# 51单片机初级入门实战教程

51DANPIANJI
CHUJI RUMENSHIZHAN JIAOCHENG

■ 安康 徐玮 等编著






机械工业出版社
CHINA MACHINE PRESS

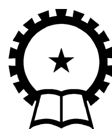
VISIT…

LANZAROTE

Caliente.com

# 51 单片机初级入门实战教程

安康　徐玮　等编著



机 械 工 业 出 版 社

本书是以最为流行的51系列单片机为知识主体，使用C语言对51单片机软件进行程序设计。全书总共分为三部分：①单片机基础知识篇；②单片机基础案例实践篇；③单片机综合案例实践篇。三部分内容逐次递进，初学者通过第一部分单片机基础知识的学习，结合第二部分能够独立设计一些简单的单片机技术案例，在第二部分实践基础上能够进一步研究和创新完成一些综合性案例。全书以案例驱动的方式，理论与实践相结合，带领读者循序渐进地完成51单片机知识的学习。本书实例丰富，图文并茂，通俗易懂，即使读者没有任何单片机知识的基础，也可以通过本书的学习让您跨入单片机世界的大门。

为了方便读者快速掌握单片机技术知识，本书的配套光盘中已含所有案例项目对应的电路图和程序代码，以及一些常用的电子系统设计开发软件。本书可作为中高等职业院校、应用型本科院校等教学用书，也可以作为单片机爱好者自学教材。

**图书在版编目（CIP）数据**

51单片机初级入门实战教程/安康等编著. —北京：机械工业出版社，2014.8
ISBN 978-7-111-47690-0

Ⅰ.①5… Ⅱ.①安… Ⅲ.①单片微型计算机-教材 Ⅳ.①TP368.1

中国版本图书馆CIP数据核字（2014）第188731号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：林春泉 责任编辑：王 荣
版式设计：赵颖喆 责任校对：任秀丽 胡艳萍
责任印制：刘 岚
北京京丰印刷厂印刷
2015年1月第1版·第1次印刷
184mm×260mm·18.75印张·451千字
0 001—3 000册
标准书号：ISBN 978-7-111-47690-0
ISBN 978-7-89405-661-0（光盘）
定价：59.00元（含1CD）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

| 电话服务 | 网络服务 |
| --- | --- |
| 社服务中心：（010）88361066 | 教材网：http://www.cmpedu.com |
| 销售一部：（010）68326294 | 机工官网：http://www.cmpbook.com |
| 销售二部：（010）88379649 | 机工官博：http://weibo.com/cmp1952 |
| 读者购书热线：（010）88379203 | **封面无防伪标均为盗版** |

# 前　言

从近几年企业对人才的需求来看，电子工程师的人才需求占据了 IT 企业人才市场非常大的一块蛋糕，电子工程师人才供不应求的同时，人才本身要求既要有经验又要有实践，单片机技术作为电子以及嵌入式项目开发的技术之一是一门应用性极强的学科，需要理论和实践有机的结合，单片机实现的电路具有体积小、元器件少、功能强、可靠性高等优点，一直都受到电子爱好者的喜爱。

本书编写着眼于“实用、适用”、“简单易懂”、“快速上手”、“举一反三”的指导思想。全书以理论与实践相结合为主线，通过案例驱动使得读者在动手实践过程中加深理论知识的学习，能够在学习过程中尽量做到反复理解和操作，最后能够独立完成技术案例，培养读者的技术创新能力。

本书讲解简单易懂，案例丰富，图文并茂，同时配套光盘中包含每个案例设计所需要的电路原理图和对应的源程序代码，使您的学习更为方便，相信即使您之前没有任何单片机知识的基础，通过阅读本书后，您一定能够运用单片机技术设计出实用的电子产品。

全书总共分为三部分内容：单片机基础知识篇、单片机基础案例实践篇、单片机综合案例实践篇。

单片机基础知识篇：考虑到 C 语言要易于阅读和理解，本书主要讲解如何利用 C 语言对单片机进行程序设计，包括 C 语言仿真环境 Keil C51 的学习以及 ISP 在线下载功能，对于初学者开始接触单片机比较关注的是单片机能做哪些事情、如何学好单片机。因此，为了能够让初学者快速步入单片机世界的大门，本书通过理论与实践相结合、以“项目案例”的方式引导初学者学习单片机的技术知识。单片机基础知识部分主要介绍了单片机的技术发展趋势以及学习方法、单片机硬件系统及体系结构（包括引脚定义、存储器、定时/计数器、中断，串行通信）等，另外还介绍了单片机采用 C 语言编程（包括 C 语言的数据结构、函数、数组和程序设计语句）等，通过基础知识的学习，使得初学者具备单片机一定的理论技能，为后面利用单片机进行案例设计做好基础。

单片机基础案例实践篇：经过第一部分单片机基础理论知识的学习，相信您对单片机的知识已经有了比较深入的了解，考虑到单片机是一门实践性极强的技术，需要读者通过具体的实践和实战对理解的理论知识进行简单的应用。单片机基础案例实践部分将为读者介绍一些简单易懂、易操作的基础案例，例如花样流水灯闪烁、按键控制、外部中断控制、数码管显示技术、定时器、串行通信、液晶显示技术和步进电动机控制等。内容讲解过程中，既介绍了案例的设计原理同时又对案例的硬件电路进行了阐述，特别在程序设计思想上，尽可能地用简洁的语言清晰阐述，让初学单片机的读者容易理解其概念和思想，有利于初学者举一反三，掌握单片机的应用为进一步独立、高效地设计复杂的电子产品做好铺垫。

单片机综合案例实践篇：经过单片机理论知识的学习以及一些单片机基础案例的锻炼，相信您对单片机进行一些实用的电子产品研发有了一定的想法，非常希望自己能够独立设计一些复杂的电子系统。在单片机综合案例实践部分，将为读者介绍一些单片机综合性案例，

让您从单片机知识学习的水平升华到产品开发的程度。在综合案例的安排上，重点突出“应用”和“实用”两个特点，包括可播音的温湿度测量系统、智能型充电器、无线遥控开关、远程果苗生长参数监测系统、电子密码锁、红外遥控电动机转速系统、智能小车寻迹系统等。通过这部分知识的学习，读者具备了初步的产品开发能力，可以独立、高效地制作一些电子系统，这时就完全踏入到单片机世界的大门里。

本书可以作为中高职院校、应用型本科院校进行单片机课程设计、毕业设计的指导教材；也可以作为初学单片机读者的参考用书，书中所涉及的案例稍加修改均可以应用在自己的工作或者用来完成自己的单片机课题，通过本书的学习使得读者能够真正掌握单片机技术，将理论知识与实践相结合，融会贯通、学以致用。

特别感谢各位同事和朋友的热心帮助，使得本书能够顺利完成。衷心盼望本书能够对从事单片机技术工作的朋友有所帮助。

参与本书编写工作的主要人员有杭州师范大学钱江学院安康、王玉槐、张慧熙、孙亚萍、李静、王李冬、王琦晖、叶霞、曹世华、丁群芳；杭州晶控电子有限公司徐玮以及浙大网新轨道交通工程有限公司安宁等，全书由安康统稿并审校。本书的编写工作获得杭州师范大学钱江学院院级重点学科“电子科学与技术”以及市级重点学科建设项目“物联网工程学科”大力支持。

由于作者水平有限，书中难免有错误与不妥之处，诚邀广大读者提出意见并不吝赐教。

**编者**

**2014 年 8 月**

# 目　　录

前言

## 第一部分　单片机基础知识篇


第1章　绪论……………………………………2
1.1　单片机技术发展趋势 ……………………2
1.2　单片机技术定义以及应用 ………………3
1.2.1　单片机技术定义 ……………………3
1.2.2　单片机应用 …………………………3
1.3　单片机开发板简介 ………………………5
1.4　单片机学习方法 …………………………6
1.5　本章小结 …………………………………6
第2章　Keil C51 软件开发环境与 ISP 在线下载 ……………………………7
2.1　Keil C51 μVision4 软件介绍 ……………7
2.2　Keil C51 μVision4 软件安装与卸载 ……………………………………7
2.2.1　Keil C51 μVision4 软件安装 ………7
2.2.2　Keil C51 μVision4 软件卸载 ………11
2.3　Keil C51 μVision4 软件操作流程 ………13
2.3.1　Keil C51 操作界面 …………………13
2.3.2　Keil C51 工程创建应用 ……………18
2.4　ISP 在线下载操作 ………………………26
2.5　本章小结……………………………………29
第3章　51 单片机硬件系统及体系结构 …………………………………30
3.1　单片机基本结构与引脚功能……………30
3.1.1　单片机基本结构………………………30
3.1.2　单片机引脚功能………………………33
3.2　单片机存储器………………………………35
3.2.1　程序存储器……………………………35
3.2.2　数据存储器……………………………35
3.3　单片机中断系统……………………………37
3.3.1　中断定义………………………………37
3.3.2　中断系统概述…………………………38
3.3.3　中断控制………………………………38
3.3.4　中断处理………………………………41
3.4　单片机定时/计数系统 ……………………42
3.4.1　定时/计数器结构及工作原理………………………………………42
3.4.2　定时/计数器特殊控制寄存器 TMOD、TCON ……………………42
3.4.3　定时/计数器工作方式 ………………44
3.5　单片机串行通信系统………………………47
3.5.1　串行通信结构与原理…………………47
3.5.2　串行控制与状态寄存器………………47
3.5.3　串行通信工作方式……………………49
3.5.4　波特率设置……………………………50
3.6　本章小结……………………………………51
第4章　51 单片机 C 语言程序设计 ……………………………………52
4.1　C 语言简介 …………………………………52
4.2　数据结构……………………………………53
4.2.1　数据类型………………………………53
4.2.2　常量与变量……………………………54
4.3　运算符与表达式……………………………56
4.3.1　运算符分类……………………………56
4.3.2　算术运算符与表达式…………………56
4.3.3　关系运算符与表达式…………………57
4.3.4　逻辑运算符和表达式…………………57
4.3.5　赋值运算符和表达式…………………57
4.3.6　位运算符与表达式……………………58
4.4　函数使用……………………………………58
4.4.1　C 语言程序的基本结构 ……………58
4.4.2　函数定义 ……………………………59
4.4.3　函数调用 ……………………………60
4.4.4　函数的嵌套调用和递归调用………61
4.5　数组与指针…………………………………63
4.5.1　数组 …………………………………63
4.5.2　指针 …………………………………64
4.6　程序设计语句………………………………66

4.6.1 选择语句 …… 67
4.6.2 循环语句 …… 69
4.6.3 转移语句 …… 71
4.7 本章小结 …… 72

## 第二部分 单片机基础案例实践篇

**第 5 章 单个 LED 点亮项目** …… 74
5.1 项目需求 …… 74
5.2 项目工作原理分析 …… 74
5.3 项目硬件电路设计 …… 74
5.4 项目软件程序设计 …… 75
5.5 系统调试结果总结 …… 76
**第 6 章 花样流水灯闪烁项目** …… 78
6.1 项目需求 …… 78
6.2 项目工作原理分析 …… 78
6.3 项目硬件电路设计 …… 79
6.4 项目软件程序设计 …… 80
6.5 系统调试结果总结 …… 82
**第 7 章 单片机独立按键控制项目** …… 83
7.1 项目需求 …… 83
7.2 项目工作原理分析 …… 83
7.3 项目硬件电路设计 …… 83
7.4 项目软件程序设计 …… 85
7.5 系统调试结果总结 …… 88
**第 8 章 单片机外部中断控制项目** …… 89
8.1 项目需求 …… 89
8.2 项目工作原理分析 …… 89
8.3 项目硬件电路设计 …… 90
8.4 项目软件程序设计 …… 91
8.5 系统调试结果总结 …… 92
**第 9 章 数码显示技术项目** …… 94
9.1 项目需求 …… 94
9.2 项目工作原理分析 …… 94
9.3 项目硬件电路设计 …… 96
9.4 项目软件程序设计 …… 97
9.4.1 数码管静态显示 …… 97
9.4.2 数码管动态显示 …… 99
9.5 系统调试结果总结 …… 101
9.5.1 数码管静态显示调试结果 …… 101
9.5.2 数码管动态扫描显示调试结果 …… 101
**第 10 章 单片机定时控制项目** …… 102
10.1 项目需求 …… 102
10.2 项目工作原理分析 …… 102
10.3 项目硬件电路设计 …… 103
10.4 项目软件程序设计 …… 104
10.5 系统调试结果总结 …… 106
**第 11 章 单片机控制蜂鸣器项目** …… 108
11.1 项目需求 …… 108
11.2 项目工作原理分析 …… 108
11.3 项目硬件电路设计 …… 108
11.4 项目软件程序设计 …… 109
11.5 系统调试结果总结 …… 112
**第 12 章 单片机串口通信项目** …… 113
12.1 项目需求 …… 113
12.2 项目工作原理分析 …… 113
12.3 项目硬件电路设计 …… 114
12.4 项目软件程序设计 …… 117
12.5 系统调试结果总结 …… 120
**第 13 章 单片机实现 4×4 矩阵键盘控制项目** …… 122
13.1 项目需求 …… 122
13.2 项目工作原理分析 …… 122
13.3 项目硬件电路设计 …… 124
13.4 项目软件程序设计 …… 126
13.5 系统调试结果总结 …… 129
**第 14 章 单片机实现字符型液晶显示项目** …… 130
14.1 项目需求 …… 130
14.2 项目工作原理 …… 130
14.3 项目硬件电路设计 …… 133
14.4 项目软件设计 …… 133
14.5 项目调试 …… 137
**第 15 章 单片机实现步进电动机控制项目** …… 138
15.1 项目需求 …… 138
15.2 项目工作原理分析 …… 138
15.3 项目硬件电路设计 …… 140
15.4 项目软件设计 …… 143
15.5 项目调试 …… 145

## 第三部分　单片机综合案例实践篇


第 16 章　家用温湿度测量播报系统设计 …… 148

16.1　项目背景和设计意义 …… 148

16.1.1　项目背景 …… 148

16.1.2　项目设计意义 …… 149

16.2　项目方案论证和方案选择 …… 149

16.2.1　项目方案论证 …… 149

16.2.2　设计方案选择 …… 149

16.3　家用温湿度测量播报系统原理及功能 …… 150

16.3.1　家用温湿度测量播报系统工作原理 …… 150

16.3.2　家用温湿度测量播报系统功能分析 …… 150

16.4　家用温湿度测量播报系统硬件电路设计 …… 151

16.4.1　单片机最小系统模块设计 …… 151

16.4.2　温湿采集模块设计 …… 152

16.4.3　液晶显示模块设计 …… 154

16.4.4　语音播报模块设计 …… 155

16.5　家用温湿度测量播报系统的软件实现 …… 158

16.5.1　单片机控制主程序软件设计 …… 159

16.5.2　温湿采集程序设计 …… 160

16.5.3　LCD 显示程序设计 …… 161

16.5.4　语音播报程序设计 …… 162

16.6　系统调试 …… 163

16.6.1　软件调试 …… 163

16.6.2　实物调试中遇到问题 …… 164

16.7　总结 …… 165

附件：设计的电路原理图 …… 166

第 17 章　单片机实现智能充电器设计 …… 167

17.1　项目背景和设计意义 …… 167

17.1.1　项目背景 …… 167

17.1.2　设计意义 …… 168

17.2　设计总体方案 …… 168

17.3　智能充电器实现原理及功能 …… 169

17.3.1　智能充电器实现原理 …… 169

17.3.2　智能充电器的功能分析 …… 170

17.4　智能充电器硬件电路设计 …… 170

17.4.1　单片机最小系统设计 …… 170

17.4.2　充电控制模块设计 …… 172

17.4.3　供电电压模块 …… 173

17.5　智能充电器软件实现 …… 174

17.5.1　单片机控制主程序设计 …… 174

17.5.2　充电控制程序 …… 175

17.5.3　串口发送数据 …… 176

17.6　系统调试和结果分析 …… 177

17.6.1　电路原理图设计 …… 177

17.6.2　程序调试 …… 177

17.6.3　程序下载 …… 178

17.6.4　结果分析 …… 178

17.6.5　系统调试中所遇到问题 …… 178

17.7　总结 …… 180

附件：设计的电路原理图 …… 180

第 18 章　无线遥控开关系统设计 …… 182

18.1　项目背景及意义 …… 182

18.1.1　项目背景 …… 182

18.1.2　设计意义 …… 182

18.2　方案论证 …… 182

18.2.1　设计方案一 …… 182

18.2.2　设计方案二 …… 183

18.2.3　方案比较与选择 …… 183

18.3　无线遥控开关系统概述 …… 184

18.3.1　工作原理 …… 184

18.3.2　功能分析 …… 184

18.4　无线遥控开关系统硬件设计 …… 185

18.4.1　发射模块 …… 185

18.4.2　无线遥控开关电路设计 …… 187

18.5　无线遥控开关软件设计 …… 192

18.5.1　开关无线接收程序设计 …… 192

18.5.2　数码显示程序设计 …… 193

18.6　系统调试 …… 194

18.6.1　程序编译 …… 194

18.6.2　程序下载 …… 195

18.6.3　调试出现的问题 …… 195

18.7　总结 …… 196

附件：设计的电路原理图 …… 196
**第 19 章 融合物联感知与 GSM 的果园环境监测系统设计** …… 198
19.1 项目说明 …… 198
19.1.1 研究背景 …… 198
19.1.2 研究现状 …… 198
19.1.3 研究内容 …… 199
19.2 果园环境监测系统方案设计 …… 199
19.2.1 系统结构原理 …… 199
19.2.2 系统功能分析 …… 201
19.3 果园环境远程监测系统电路设计 …… 201
19.3.1 单片机最小系统 …… 201
19.3.2 现场端采集电路 …… 202
19.3.3 GSM TC35i 外围电路设计 …… 204
19.4 果园环境监测系统现场感知端软件实现 …… 207
19.4.1 主程序设计 …… 207
19.4.2 现场端数据信息发送程序设计 …… 208
19.4.3 现场端数据信息接收程序设计 …… 209
19.5 系统测试 …… 210
19.5.1 系统测试步骤 …… 210
19.5.2 测试结果分析 …… 211
19.6 结论 …… 211
附件：果园现场数据采集端电路原理图 …… 212
**第 20 章 单片机实现电子密码锁设计** …… 214
20.1 项目说明 …… 214
20.1.1 项目背景 …… 214
20.1.2 电子密码锁优点 …… 215
20.1.3 研究内容 …… 215
20.2 系统总体设计 …… 216
20.2.1 系统工作原理 …… 216
20.2.2 系统结构 …… 216
20.3 系统硬件电路设计 …… 217
20.3.1 AT89S52 单片机最小系统设计 …… 217
20.3.2 密码存储电路设计 …… 218
20.3.3 4×4 矩阵键盘模块设计 …… 219
20.3.4 数码管显示电路设计 …… 220
20.3.5 报警指示模块设计 …… 220
20.3.6 电源模块电路设计 …… 221
20.4 系统软件程序设计 …… 222
20.4.1 主程序设计 …… 222
20.4.2 串行 EEPROM 读写程序设计 …… 223
20.4.3 4×4 矩阵键盘处理程序设计 …… 226
20.5 系统调试总结 …… 227
附件：系统设计的电路原理图 …… 227
**第 21 章 红外遥控电动机转速系统设计** …… 229
21.1 项目说明 …… 229
21.1.1 研究背景 …… 229
21.1.2 研究内容 …… 230
21.2 系统总体设计 …… 230
21.2.1 系统结构 …… 230
21.2.2 红外遥控器工作原理 …… 230
21.2.3 步进电动机工作原理 …… 232
21.3 系统硬件电路设计 …… 233
21.3.1 单片机最小系统设计 …… 233
21.3.2 红外遥控器模块设计 …… 234
21.3.3 步进电动机模块设计 …… 234
21.3.4 LCD 显示模块设计 …… 235
21.4 系统软件程序设计 …… 236
21.4.1 主程序设计 …… 236
21.4.2 红外遥控器解码程序设计 …… 236
21.4.3 LCD 显示程序 …… 238
21.4.4 步进电动机控制程序 …… 239
21.5 系统调试总结 …… 241
附件：系统设计的电路原理图 …… 241
**第 22 章 智能小车自动寻迹系统设计** …… 242
22.1 项目背景和研究内容 …… 242
22.1.1 项目背景 …… 242
22.1.2 研究内容 …… 242
22.1.3 系统设计技术 …… 242
22.2 系统电路设计 …… 244
22.2.1 系统工作原理 …… 244
22.2.2 系统硬件电路设计 …… 244
22.3 系统软件设计 …… 250
22.3.1 主程序设计 …… 250
22.3.2 无线发射程序设计 …… 251

22.3.3　数码管动态显示程序设计 ……… 252
22.4　系统调试 …………………………… 252
附件：系统设计的电路原理图 ………… 253
**第 23 章　红外遥控风扇控制系统设计** ……… 256
23.1　项目说明 …………………………… 256
23.1.1　研究背景 ……………………… 256
23.1.2　研究方案 ……………………… 256
23.2　系统概述 …………………………… 257
23.3　系统硬件电路设计 ………………… 257
23.3.1　AT89S52 单片机最小系统设计 ……… 257
23.3.2　温度传感器电路设计 ………… 259
23.3.3　LCD1602 显示模块设计 ……… 262
23.3.4　红外接收模块 ………………… 263
23.3.5　电动机驱动模块设计 ………… 263
23.3.6　存储电路 ……………………… 264
23.4　系统软件设计 ……………………… 265
23.4.1　主程序设计 …………………… 265
23.4.2　温度采集子程序设计 ………… 266
23.4.3　红外接收程序设计 …………… 267
23.4.4　LCD 显示子程序设计 ………… 268
23.4.5　电动机驱动子程序设计 ……… 269
23.4.6　温度存储子程序设计 ………… 270
23.5　系统调试与总结 …………………… 272
23.5.1　系统调试 ……………………… 272
23.5.2　系统总结 ……………………… 273
附件：系统设计的电路原理图 ………… 273
**第 24 章　多功能微电脑模拟电子秤设计** ……… 275
24.1　项目说明 …………………………… 275
24.1.1　项目背景 ……………………… 275
24.1.2　设计总体方案论证 …………… 275
24.2　多功能微电脑电子秤实现原理 …… 276
24.3　微电脑电子秤硬件电路设计 ……… 276
24.3.1　51 单片机最小系统 …………… 276
24.3.2　键盘电路 ……………………… 278
24.3.3　ADC0809 接口电路 …………… 279
24.3.4　数码显示电路 ………………… 279
24.4　微电脑电子秤软件实现 …………… 281
24.4.1　主程序设计 …………………… 281
24.4.2　键盘控制程序设计 …………… 282
24.4.3　显示程序设计 ………………… 284
24.5　系统调试总结 ……………………… 285
附件：系统设计的电路原理图 ………… 285
**参考文献** ……………………………… 288

# 第一部分　单片机基础知识篇

单片机基础知识部分主要向读者介绍单片机技术背景和发展趋势，单片机硬件体系结构以及C语言的学习，包括使用C语言对单片机进行软件开发、Keil C51开发环境的学习和调试。对于初学者通过第一部分单片机基础知识的学习，使初学者快速掌握单片机的基本技能，为后面进一步学习单片机的应用做好基础工作。单片机基础知识部分由4章内容构成。

第1章　绪论
第2章　Keil C51软件开发环境与ISP在线下载
第3章　51单片机硬件系统及体系结构
第4章　51单片机C语言程序设计

# 第1章　绪　　论

## 1.1　单片机技术发展趋势

单片机诞生于20世纪70年代末，经历了SCM、MCU、SoC三大阶段。单片微型计算机（Single Chip Microcomputer，SCM）阶段，主要是寻求最佳的单片形态嵌入式系统的最佳体系结构。“创新模式”获得成功，奠定了SCM与通用计算机完全不同的发展道路。微控制器（Micro Controller Unit，MCU）阶段，主要技术发展方向是不断扩展满足嵌入式应用的同时，对系统要求的各种外围电路与接口电路，突显其对系统的智能化控制能力。单片机是嵌入式系统的独立发展之路，向MCU阶段发展的重要因素，就是寻求应用系统在芯片上的最大化解决；因此，专用单片机的发展自然形成了SoC化趋势，详细的发展阶段如下：

1974年12月，美国仙童（Fairchild）公司推出了世界上第一台8位单片机F8。单片机的发展过程分为以下几个发展阶段。

第一代单片机（1974~1976年）：

单片机发展的起步阶段。集成度也较低，并且采用了双片形式。代表产品有Fairchild公司的F8和Mostek公司的3870等。

第二代单片机（1976~1978年）：

这是单片机的发展阶段。最典型的产品有Intel公司的MCS-48系列单片机。

第三代单片机（1979~1982年）：

这是8位单片机的成熟阶段。代表产品有Intel公司的MCS-51系列机、Motorola公司的MC6801系列机、Zilog公司的Z8系列机等。

第四代单片机（1983年以后）：

1983年以后是16位单片机和8位高性能单片机并行发展的时代。随着微电子技术、IC设计、EDA工具的发展，基于SoC的单片机应用系统设计会有较大的发展。因此，对单片机的理解可以从单片微型计算机、单片微控制器延伸到单片应用系统。

目前，单片机正朝着多功能、多选择、高速度、低功耗、低价格、扩大存储容量和加强I/O功能以及结构兼容方向发展，单片机的发展趋势具体体现在以下4个方面：

（1）多功能　在单片机中尽可能多的将应用系统中所需要的存储器、各种功能的I/O口都集成在一块芯片内，即外围器件内装化，如把LED、LCD和VFD显示驱动器集成在单片机中，如把A-D、D-A以及多路模拟开关和采样/保持器也集成在单片机中。

（2）高性能　精简指令集计算机（Reduced Instruction Set Computer，RISC）是计算机中央处理器的一种设计模式。使用RISC体系结构、并行流水线操作和DSP等设计技术，使单片机的指令运行速度得到大大提高，其电磁兼容等性能明显优于同类型的微处理器。

（3）全盘CMOS化　单片机采用两种半导体工艺生产，HMOS工艺即高密度短沟道MOS工艺；CHMOS工艺即互补金属氧化物的HMOS工艺，如8051的功耗为630mW，而

80C51 的功耗仅为 120mW。从第三代单片机起开始淘汰非 CMOS 工艺。

（4）推行串行扩展总线　显著减少引脚数量，简化系统结构。随着外围器件串行接口的发展，单片机串行接口的普遍化、高速化使得并行扩展接口技术日渐衰退。推出了删去并行总线的非总线单片机，需要外扩器件（存储器、I/O 等），采用串行扩展总线，甚至用软件虚拟串行总线来实现。

另外单片机具体的功能体现在如下几个方面：

（1）4 位、8 位、16 位、32 位单片机共存，并各有自己的生存空间。

（2）CPU 功能不断增强、运行速度不断提高。

（3）内部资源增多，增加存储器容量、片内外设如 A-D、D-A、LED/LCD 驱动、PWM 等。

（4）引脚的多功能化。

（5）低电压和低功耗。

（6）结合 ASIC 和 RISC 技术，使单片机的应用范围进一步扩大。

## 1.2　单片机技术定义以及应用

### 1.2.1　单片机技术定义

单片机的定义：采用超大规模集成电路技术把具有数据处理能力的中央处理器（CPU）、随机存储器（RAM）、只读存储器（ROM）、多种 I/O 口、中断系统、定时器/计时器等功能（可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A-D 转换器等电路）集成到一块硅片上构成的一个小而完善的计算机系统。

单片机的类型和型号比较多，目前使用比较广的单片机有 MCS-51 系列、AVR、PIC、MSP430 等。而 51 系列是应用最广泛的，也是最容易入门的、最有代表性的，常用的型号有 STC89XXX、AT89XXX、P89XXX 等。51 系列都采用 8051 内核，因此不同厂家的 51 单片机几乎互相兼容。由于 STC 单片机具备诸多优点，因此市场份额最大，用得最广。AVR 单片机的速度比较快，性能比 51 单片机高，但价格也较高。各种类型单片机都是相通的，只要学好任何一种类型单片机，其他单片机的学习通过芯片使用手册可以做到举一反三，掌握速度也是非常快的。

### 1.2.2　单片机应用

随着近几年 IT 技术的迅猛发展，使得 IT 产业在工业、农业、国防科研及日常生活各个领域均显示了日益旺盛的生命力。在国内，20 多年来，微型机不断地更新换代，新的产品层出不穷。在微机的大家族中，近年来单片微型计算机以其低价位、高性能的特点异军突起，发展极为迅速，应用十分广泛。目前，单片机技术已经普及到我们生活、工作、科研等各个领域，已经发展成为一种比较成熟的技术。而单片机的应用提高了机电设备的技术水平和自动化程度，对各行各业的技术改造和产品更新换代起到了重要的推动作用。

**1. 单片机特别适用于机、电、仪一体的智能化产品**

在各类仪器仪表中（包括医疗器械、色谱仪、温度、湿度、流量、流速、电压、频率、

功率、长度、硬度、元素测定等）引入单片机，使仪器仪表数字化、智能化、微型化等功能大大提高。

**2. 单片机在工业控制中的应用**

**3. 单片机在通信方面的应用**

单片机成功地应用于玩具、游戏机、充电器、按摩器、IC 卡电话、IC 卡水表、IC 卡煤气表、IC 卡电能表、流量温控仪表、家庭自动化、电子锁、电子秤、步进电机、防盗报警、电子日历时钟等日常生活的产品中。

**4. 计算机外部连接设备**

图形终端、彩色黑白复印机、软盘及硬盘驱动器、磁带机、打印机的内部都采用单片机进行控制。

不难发现单片机是一个万能器件，它可以完成很多设备的控制工作，尤其对于初学者很想以最快的速度学会单片机并设计一块由单片机控制的电子系统，这里给读者展示一些由单片机完成的一些简单的电子系统供初学者对单片机后续的学习有个基础的认识。

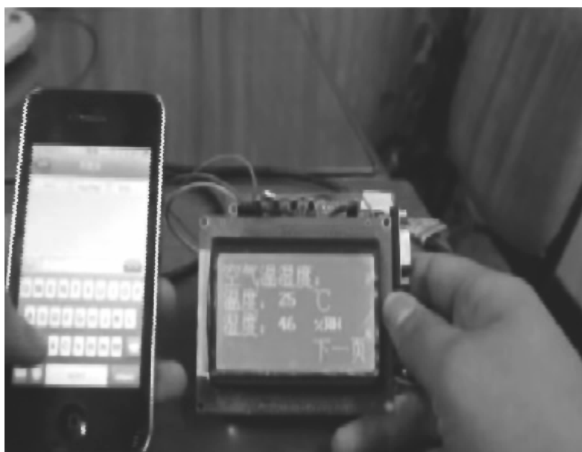


图 1-1　无线温湿度采集终端

图 1-1 是一个用单片机控制 GSM TC35i 实现远程空气和土壤温湿度监测系统，针对传统果苗生长环境信息获取科学度低、时效性差等不足，将物联感知与 GSM 技术应用到果苗生长环境监测系统中。通过对果苗生长所需的空气和土壤温湿度信息进行采集，利用 GSM TC35i 模块，以短消息的方式实现数据远程传输，实时将采集的信息发送到果农手机端，有利于减轻果农劳动强度、提高果苗生长品质，这是一个应用于农业生产的案例。

图 1-2 是一个利用单片机技术采用语音芯片 ISD1420 实现温湿采集播报系统，系统硬件电路主要由单片机最小系统模块、液晶显示模块、温度传感器模块、语音芯片模块构成；软件设计在 keil 仿真环境下采用 C 语言编程。设计的温湿播报系统能够在液晶屏上显示测量的温度和湿度，同时可以语音播报，实用性和可靠性强。这是一个应用日常生活或者环境监测方面的案例。

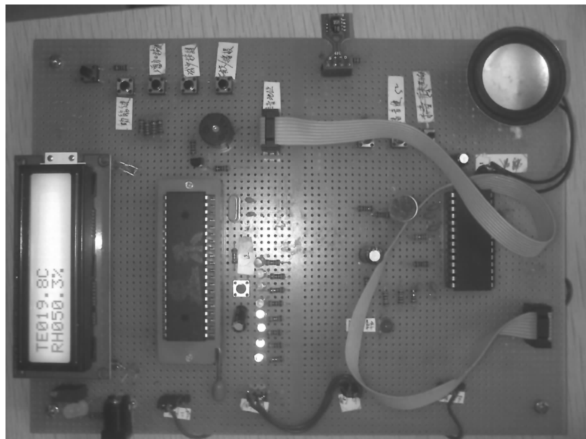


图 1-2　家用温湿测量播报系统实物图

图 1-3 是一个采用单片机利用无线通信技术设计的一款两通道无线遥控开关系统，系统硬件由发射模块和接收模

块两部分构成，其中发射模块采用 PT2262 进行编码发射，接收模块由 PT2272 解码电路、电源电路、数码显示电路和继电器控制电路构成；软件系统采用 C 语言编程，实现两通道遥控开关系统，这是一个应用于无线通信方面的案例。

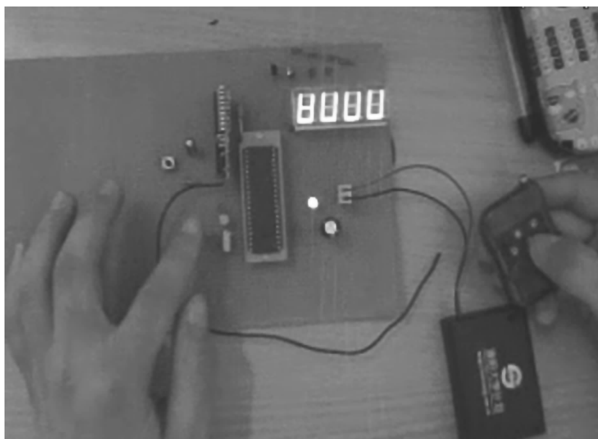


图 1-3　无线遥控开关系统设计

图 1-4 是一个利用单片机采用 MAX1898 充电芯片的一款智能充电器系统。系统硬件电路主要有单片机最小系统、充电控制模块、供电电压模块和报警模块构成；软件系统利用单片机 T0 定时器，采用 C 语言编程。系统具有预充、充电保护、自动断电和充电完成报警提示功能，避免了由于过电压充电对电池造成的损害，保护电池。这是一个应用于智能化仪器方面的案例。

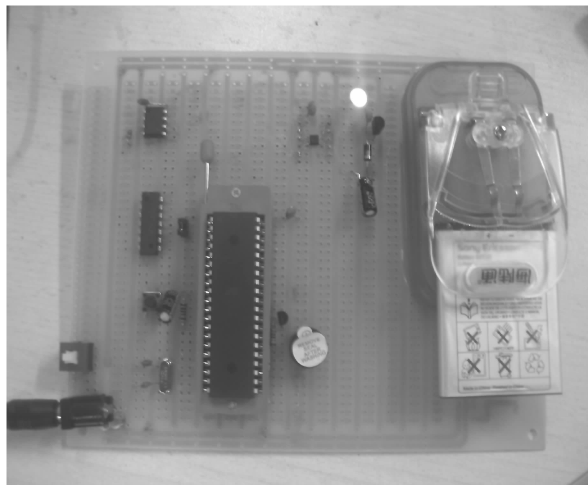


图 1-4　智能充电器的实物图

从上述 4 个简单的案例介绍中看出，单片机的应用非常广泛，同时学好单片机可以解决很多电子系统的控制问题，所以要求读者只要静下心来慢慢地将整本书仔细阅读，你就会发现单片机技术一点都不难，你一定可以设计出一套符合设计要求的单片机控制系统。

## 1.3　单片机开发板简介

单片机初学者手上最好有一块单片机开发板，这样才能将学到的单片机知识理论与实践相结合，才能真正学好单片机。一般单片机的开发板价格不是很贵，以 200 ~ 300 元的单片机开发板最为适宜，只要单片机开发板功能强大，可以帮助初学者学习单片机的外围电路，

熟悉单片机各种硬件电路和锻炼单片机软件编程。另外，开发板所附带的文件资料非常重要，开发板附赠资料至少包括整块开发板的电路原理图（原理图方便设计者读懂电路并进行二次开发和程序设计使用），还需有丰富的实例程序和各种开发软件，这样对于初学者入门单片机学习速度会加快。本书第二部分单片机基础案例实践篇中，单片机案例调试均在广州研展电子科技有限公司的YZ200单片机开发板和杭州晶控电子有限公司的51单片机综合开发板上进行调试。这里介绍一般51开发板系统的需求和系统主要特点如下：

**1. 单片机学习系统需求**

系统软硬件需求：

计算机一台（奔腾级以上的家用电脑即可，要求不苛刻）；51单片机开发板一套

Windows98/ME/XP/2003/7操作系统，最小硬盘空间为80MB。

主要硬件接口功能说明：

RS232串口：用于仿真操作（如无串口，可以用USB转RS232串口线）。

USB口：提供51单片机系统电源。

**2. 51单片机开发板系统特点**

1）编程、实验、仿真功能，具有40引脚和20引脚外扩仿真接口。

2）串口通信，支持USB转RS232串口线，可以直接用于只有USB口的便捷式计算机或台式计算机。

3）配有40Pin外接仿真头，可以作为一台独立的51单片机硬件仿真器使用，通过Keil软件配合，即可对外部硬件以及板上资源实现单步调试，设置断点，全速执行等功能。

4）开放性设计，可扩接任意功能的外围模块，如温度传感模块、语音录音芯片模块等。

## 1.4 单片机学习方法

学好单片机最有效的方法与途径关键在于是否将理论与实践相结合，多看书，多动手，多实践，这样才能将学到的理论知识进行深刻地理解与掌握。

（1）多看书：学好单片机的基本理论，对硬件系统和体系结构要有一定的了解。

（2）多动手、多实践：单片机是一门非常实用的课程，只看书是绝不能学好单片机的，要做到理论与实践并重。最好有一块功能比较全的开发板，通过在开发板上多编程，多做实验，只有不断训练，才能深入学习好单片机。

本书突破传统教科书“教条式”的学习模式，通过案例驱动采用理论与实践结合方式的学习模式，遵循由浅入深、删繁就简、理论联系实际的原则，使初学者可以在很短的学习周期内对单片机基础知识及应用快速掌握，快速进入单片机世界的大门。

## 1.5 本章小结

本章主要对单片机技术进行概述，主要从单片机技术的发展状况及应用两个方面进行阐述，读者通过阅读一些单片机开发案例了解单片机，最后针对初学者如何在较短的学习周期内掌握单片机给出一些学习意见，为后续单片机知识的学习打好基础。

# 第 2 章　Keil C51 软件开发环境与 ISP 在线下载

## 2.1　Keil C51 μVision4 软件介绍

51 单片机的开发除了需要硬件的支持，同样需要软件的支持，CPU 执行的是机器码，而用汇编语言或者 C 语言等高级语言编写的程序必须转换为机器码才能被 CPU 执行，通过编译软件如 Keil C51 编译器可以把用 C 或者汇编编写的源程序转换为机器码供 CPU 执行。本章重点介绍了比较实用的 Keil C51 编译器。Keil C51 软件是众多单片机应用开发的优秀软件之一，它集编辑、编译、仿真于一体，支持汇编、C 语言的程序设计，界面友好，易学易用。图 2-1 为启动 Keil C51 V9.00 版本既 μV4 的屏幕图。




图 2-1　启动 Keil C51 的屏幕图

## 2.2　Keil C51 μVision4 软件安装与卸载

### 2.2.1　Keil C51 μVision4 软件安装

通过网络下载 Keil C51 μVision4 的安装软件 C51V901.exe 以及许可号生成器软件 KEIL_Lic.exe，若 PC 机安装的操作系统为 win2003/xp/2000/9x，则可以直接双击 C51V901.exe 安装软件，若操作系统为 VISTA/win7 系统则选择以管理员身份运行，操作过程如图 2-2 所示。




图 2-2　VISTA/win7 系统下运行 C51 μVision4

若软件安装后无法注册，则可以在开机运行进入系统之前按住 F8 按键，进入系统安全模式，在安全模式下安装和注册 Keil 软件，软件安装和注册过程如下：

双击运行后进入 Keil C51 μVision4 编译器的安装界面如图 2-3 所示，鼠标单击 next 进入。

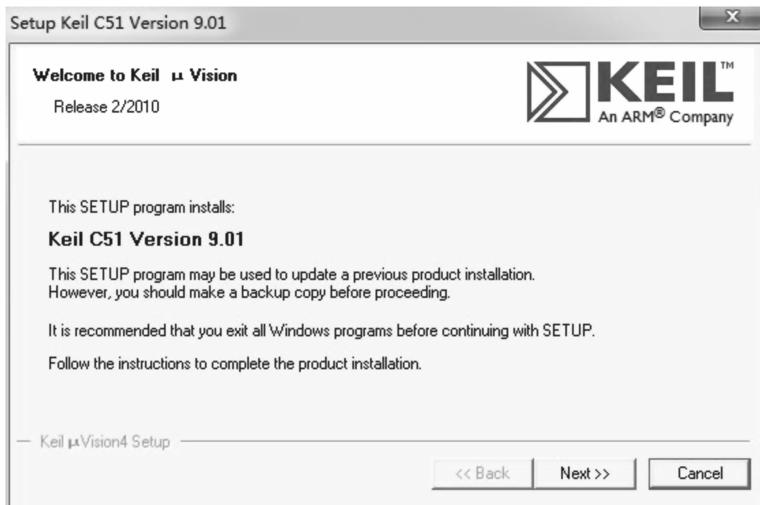



图 2-3 Keil C51 µVision4 安装界面（1）

软件安装协议阅读界面如图 2-4 所示。若阅读后无异议，鼠标选择 I agree to all the terms of the preceding License Agreement，单击 next，进入到软件路径安装界面如图 2-5 所示。




图 2-4 Keil C51 µVision4 安装界面（2）

通过 Browse 可以任意选择软件在 PC 上的安装路径如 D：\ changshikeilc51，系统默认路径为 C：\ Keil，鼠标单击 next 进入客户信息输入界面如图 2-6 所示。

需要在 first name、last name、company name 以及 E-mail 中输入相应的信息，输入完成后鼠标单击 next，进入 Keil C51 µVision4 系统安装进程状态如图 2-7 所示。

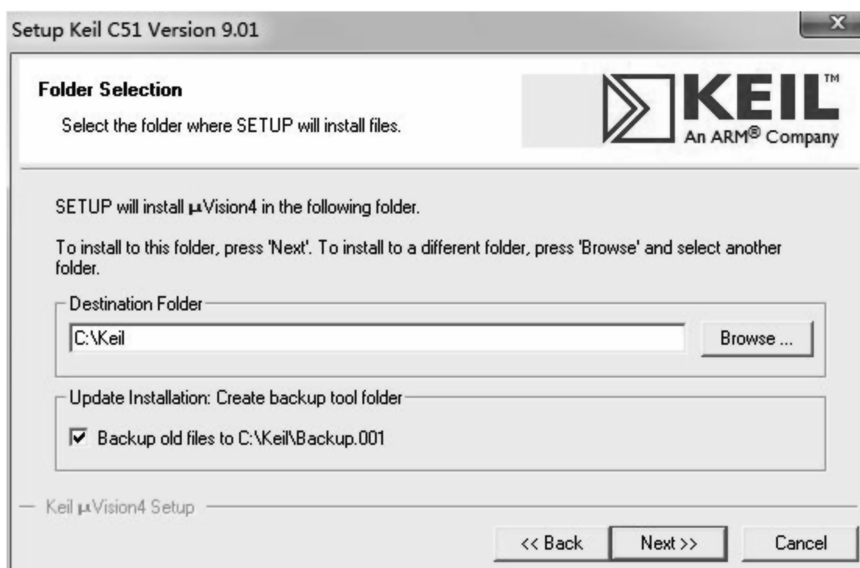



图 2-5　Keil C51 µVision4 安装界面（3）

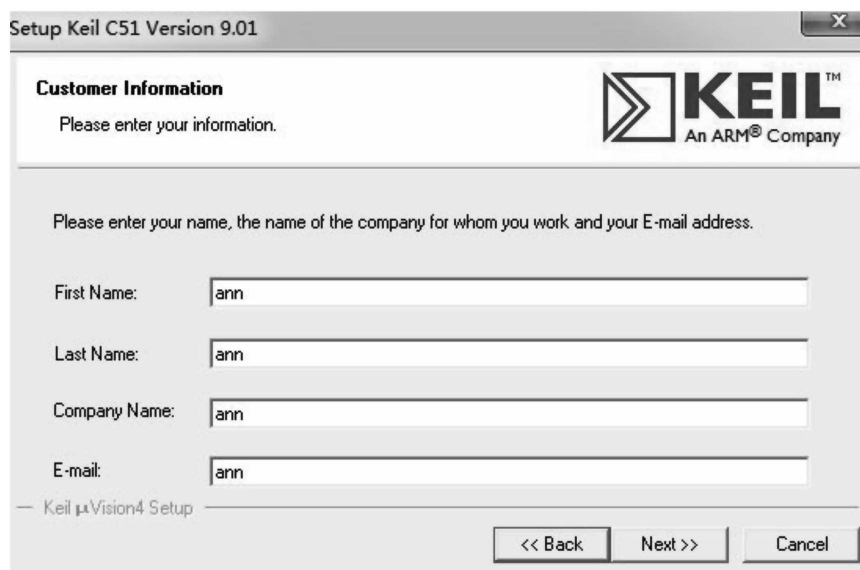



图 2-6　Keil C51 µVision4 安装界面（4）

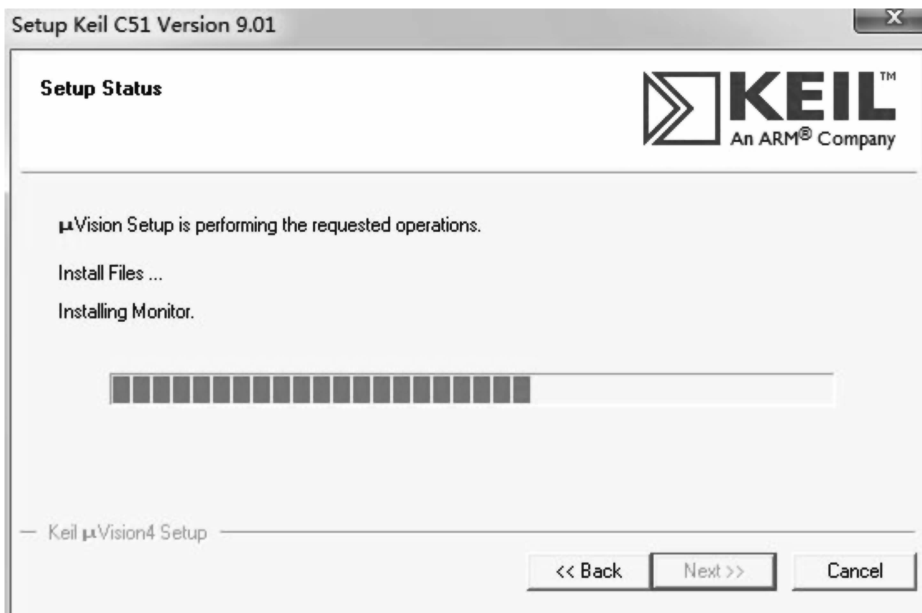



图2-7 Keil C51 µVision4 安装界面（5）

等待大约1min左右，软件安装完成进入如图2-8所示界面。

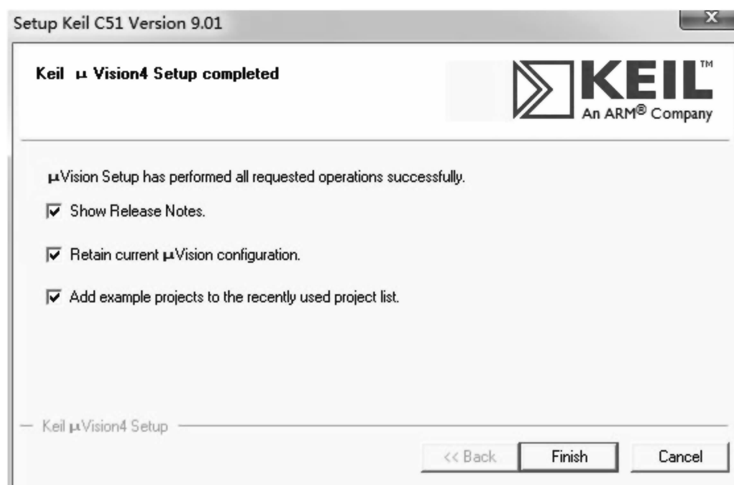



图2-8 Keil C51 µVision4 安装界面（6）

安装完成鼠标单击Finish，进入到Keil C51 µVision4软件编译界面如图2-9所示。

为了保证软件能够正确编译和仿真，需要对软件进行许可号认证，在图2-9界面下单击File→License Management，则弹出License Management管理窗口如图2-10所示，同时打开KEIL_Lic. exe许可序列号生成软件，将computer ID（每台机器的CID是不一样）号码复制到Keil序列生成器窗口如图2-11中的CID区域。

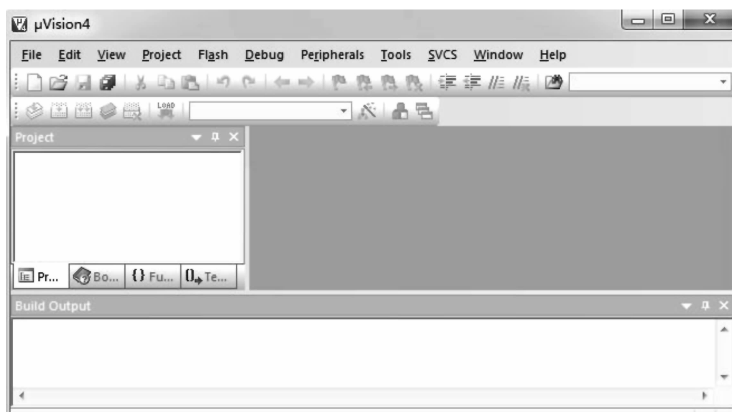



图 2-9　Keil C51 µVision4 软件工作界面

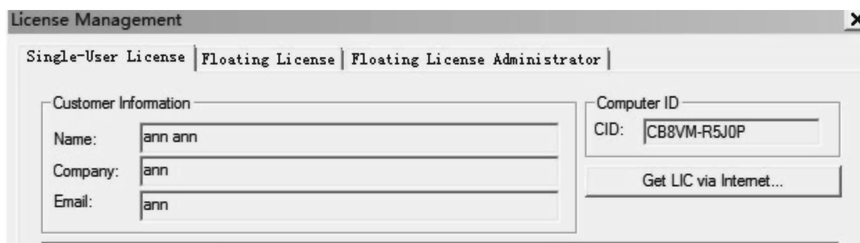



图 2-10　License Management 管理窗口

鼠标单击 Generate 会在其上方生成一串 Keil 许可号序列，将该序列复制粘贴到图 2-12 License Management 管理窗口中的 New License ID（LCD）中，单击右边的 Add LIC，若在上方的 Product 显示的是 PK51 Prof Developers Kit，则 Keil 软件注册成功。若软件使用过程中超过有效期，则可以多次生成许可号重新注册使用 Keil 编译软件。

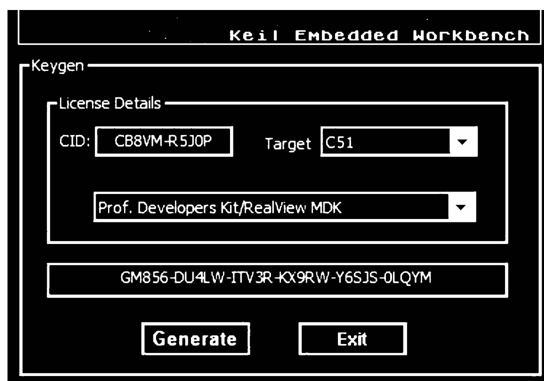



图 2-11　Keil 序列生成器窗口

## 2.2.2　Keil C51 µVision4 软件卸载

Keil 软件卸载比较简单，在 Windows 7 系统下通过程序开始菜单找到系统的控制面板，在控制面板中找到程序和功能选项如图 2-13 所示。鼠标单击程序和功能选项，则弹出卸载和更改程序窗口如图 2-14 所示，找到 Keil µVision4 鼠标右击弹出 Keil µVision4 卸载界面如图 2-15 所示，选中 Keil C51Development Tools 鼠标单击 Remove，则 Keil µVision4 软件开始卸载，卸载完成后单击 Close，关闭 Keil 软件卸载界面。

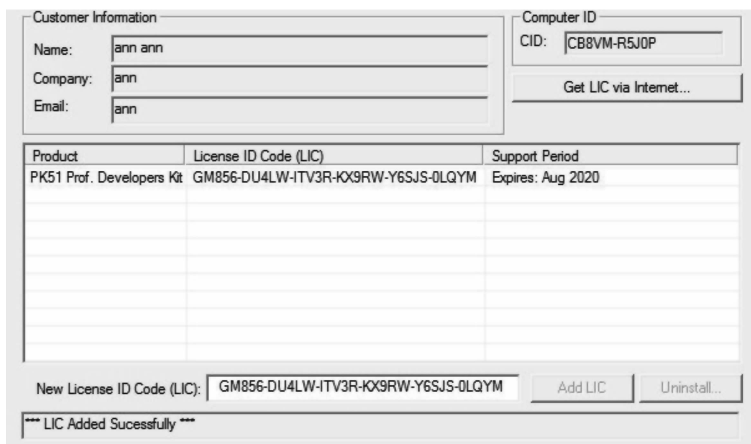



图 2-12 License Management 管理窗口




图 2-13 控制面板




图 2-14 程序卸载和更改窗口

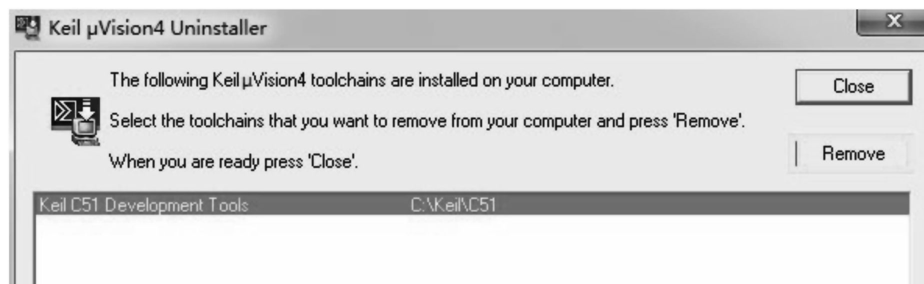



图 2-15　Keil μVision4 卸载界面

## 2.3　Keil C51 μVision4 软件操作流程

### 2.3.1　Keil C51 操作界面

Keil μVision4 安装完成后，用户可以单击运行图标 Keil uVision4 进入 Keil 编译仿真环境。Keil μVision4 软件有菜单栏，可以快速选择命令按钮的工具栏、一些源代码文件窗口、对话框窗口、信息显示窗口、编译软件允许同时打开多个源程序文件，Keil C51 工作界面如图 2-16 所示。




图 2-16　Keil C51 工作界面

**1. File 文件菜单**

鼠标单击图 2-16 工作界面菜单栏中的 File 文件菜单选项，下拉选项中的各项功能说明见表 2-1。菜单选项主要完成程序和工程文件的创建、保存和打印功能。

表 2-1　文件菜单命令说明

| File菜单 | | 描述 |
| --- | --- | --- |
| New... | Ctrl+N | 创建一个新的源文件或者文本文件 |
| Open | Ctrl+O | 打开已有的文件 |
| Close | | 关闭当前文件 |
| Save | Ctrl+S | 保存当前文件 |
| Save As... | | 保存并重命名当前文件 |
| Save All | | 保存所有打开的文件 |
| Device Database... | | 维护uVision4器件数据库 |
| License Management... | | 许可证管理 |
| Print Setup... | | 设置打印机 |
| Print... | Ctrl+P | 打印当前文件 |
| Print Preview | | 打印预览 |
| Recent Files | | 打开最近使用的文件 |
| Exit | | 退出编译环境 |

**2. Edit 编辑菜单**

鼠标单击图 2-16 工作界面菜单栏中的 Edit 编辑菜单选项，下拉选项中的各项功能说明见表 2-2。编辑选项主要完成文件的复制、粘贴、剪切、查找和替换等功能。

表 2-2　编辑菜单命令说明

| Edit编辑菜单 | | 描述 |
| --- | --- | --- |
| Undo | Ctrl+Z | 撤销上一次操作 |
| Redo | Ctrl+Y | 重做 |
| Cut | Ctrl+X | 剪切 |
| Copy | Ctrl+C | 复制 |
| Paste | Ctrl+V | 粘贴 |
| Navigate Backwards | Alt+Left | 向后导航 |
| Navigate Forwards | Alt+Right | 向前导航 |
| Insert/Remove Bookmark | Ctrl+F2 | 插入/移除书签 |
| Go to Next Bookmark | F2 | 光标移动到下一个书签 |
| Go to Previous Bookmark | Shift+F2 | 光标移动到上一个书签 |
| Clear All Bookmarks | Ctrl+Shift+F2 | 清除所有的书签 |
| Find... | Ctrl+F | 查找 |
| Replace... | Ctrl+H | 替换 |
| Find in Files... | Ctrl+Shift+F | 在几个文件中查找文字 |
| Incremental Find | Ctrl+I | 增量查找 |
| Outlining | ▸ | 提纲 |
| Advanced | ▸ | 高级编辑命令 |
| Configuration... | | 配置 |

### 3. View 视图菜单

鼠标单击图 2-16 工作界面菜单栏中的 View 视图菜单选项，下拉选项中的各项功能说明见表 2-3。视图选项主要打开各种工具栏以及功能窗口等功能。

表 2-3　编辑菜单命令说明

| 视图View | 描述 |
| --- | --- |
| Status Bar | 状态栏 |
| Toolbars | 文件工具栏 |
| Project Window | 工程窗口 |
| Books Window | 图书窗口 |
| Functions Window | 功能窗口 |
| Templates Window | 模板窗口 |
| Source Browser Window | 浏览器窗口 |
| Build Output Window | 编译输出窗口 |
| Find In Files Window | 查找文件窗口 |
| Full Screen | 全屏 |

### 4. Project 工程菜单

鼠标单击图 2-16 工作界面菜单栏中的 Project 工程菜单选项，下拉选项中的各项功能说明见表 2-4。工程菜单选项主要是创建工程、打开工程以及对工程的各种操作功能。

表 2-4　工程菜单命令说明

| 工程菜单 | 描述 |
| --- | --- |
| New µVision Project... | 创建新的工程文件 |
| New Multi-Project Workspace... | 创建工程工作空间 |
| Open Project... | 打开已有的工程文件 |
| Close Project | 关闭工程 |
| Export | 输出工程的格式 |
| Manage | 对器件 环境的管理 |
| Select Device for Target... | 从器件数据库选择一个CPU |
| Remove Item | 从工程中删除一个组或者文件 |
| Options...　Alt+F7 | 设置工具选项 |
| Clean target | 清除编译结果 |
| Build target　F7 | 转换修改过的文件并编译成应用 |
| Rebuild all target files | 重新转换所有的源文件并编译成应用 |
| Batch Build... | 通过输出的批处理文件进行编译 |
| Translate...　Ctrl+F7 | 转换当前文件 |
| Stop build | 停止编译进程 |

### 5. Debug 调试菜单

鼠标单击图 2-16 工作界面菜单栏中的 Debug 调试菜单选项，下拉选项中的各项功能说明见表 2-5。调试菜单选项主要完成对程序设置断点、单步调试等功能。

表 2-5　调试菜单命令说明

| 调试菜单 | | 描述 |
|---|---|---|
| Start/Stop Debug Session | Ctrl+F5 | 启动/停止调试模式 |
| Reset CPU | | 重启CPU |
| Run | F5 | 运行直到下一个有效的断点 |
| Stop | | 停止当前程序运行 |
| Step | F11 | 跟踪运行程序 |
| Step Over | F10 | 单步运行程序 |
| Step Out | Ctrl+F11 | 执行到当前函数的程序 |
| Run to Cursor Line | Ctrl+F10 | 运行到光标处 |
| Show Next Statement | | 显示下一条执行的语句/指令 |
| Breakpoints... | Ctrl+B | 打开断点对话框 |
| Insert/Remove Breakpoint | F9 | 在当前设置/清除断点 |
| Enable/Disable Breakpoint | Ctrl+F9 | 使能/禁能当前行的断点 |
| Disable All Breakpoints | | 禁能程序中所有断点 |
| Kill All Breakpoints | Ctrl+Shift+F9 | 清除程序中所有断点 |
| OS Support | | 系统支持 |
| Execution Profiling | | 可设置成off Time 和Call |
| Memory Map... | | 打开存储器空间配置对话框 |
| Inline Assembly... | | 对某一行重新汇编，修改汇编代码 |
| Function Editor (Open Ini File)... | | 编辑调试函数和调试配置文件 |
| Debug Settings... | | 调试设置 |

**6. Help 帮助菜单**

鼠标单击图 2-16 工作界面菜单栏中的 Help 帮助菜单选项，下拉选项中的各项功能说明见表 2-6。帮助菜单选项主要完成在使用 Keil C51 软件时遇到的问题可以寻求帮助和解决。

表 2-6　帮助菜单命令说明

| 帮助菜单 | 描述 |
|---|---|
| µVision Help | 打开帮助文件 |
| Open Books Window | 打开工程工作空间中的书签 |
| Simulated Peripherals for <Device> | 有关所选CPU的外设信息 |
| Internet Support Knowledgebase | 从网络中获得知识数据库 |
| Contact Support | 获得相关软件使用的支持 |
| Check for Update | 检测软件更新 |
| About µVision... | 显示uVision 的版本号和许可证信息 |

**7. 其他菜单选项**

在 Keil 软件操作时，鼠标单击图 2-16 工作界面菜单栏中的 Flash 菜单选项，下拉选项中的各项功能说明见表 2-7，Flash 菜单选项主要实现程序下载功能。

**表 2-7　Flash 菜单命令说明**

| Flash菜单 | 描述 |
|---|---|
| Download | 下载程序 |
| Erase | 擦除程序 |
| Configure Flash Tools... | 配置Flash工具 |

鼠标单击图 2-16 工作界面菜单栏中的 Tools 工具菜单选项，下拉选项中的各项功能说明见表 2-8 所示，Flash 菜单选项主要实现程序下载功能。

**表 2-8　Tools 菜单命令说明**

| 工具菜单 | 描述 |
|---|---|
| Set-up PC-Lint... | 配置PC-Lint |
| Lint | 在当前的编辑文件中运行PC-Lint |
| Lint All C-Source Files | 在工程的C源代码文件中运行PC-Lint |
| Customize Tools Menu... | 将用户程序加入工具菜单 |

鼠标单击图 2-16 工作界面菜单栏中的 SCVS 工具菜单选项，下拉选项中的各项功能说明见表 2-9。

**表 2-9　SCVS 菜单命令说明**

| SCVS菜单 | 描述 |
|---|---|
| Configure Software Version Control... | 配置软件版本控制系统命令 |

鼠标单击图 2-16 工作界面菜单栏中的 Window 菜单选项，下拉选项中的各项功能说明见表 2-10。

**表 2-10　Window 菜单命令说明**

| Window菜单 | 描述 |
|---|---|
| Debug Restore Views... | 调试恢复视图 |
| Reset View to Defaults | 重启默认的视图 |
| Split | 划分当前窗口为多个窗格 |
| Close All | 关闭所有窗口 |

### 2.3.2 Keil C51 工程创建应用

通过单片机与程序设计语言的学习，如 C 语言程序设计，最好的方法是直接操作实践，以下通过编写简单的 C 语言程序创建一个工程文件，对工程文件进行编译和调试引导大家学习 Keil C51 软件的基本使用方法。

（1）首先创建一个新的工程，单击 Project 菜单，在下拉菜单中选择 new μvision project 如图 2-17 所示，弹出创建工程窗口如图 2-18 所示，设定好保存创建工程的路径，在文件名中给创建工程命名如 test，扩展名为 Uvproj，鼠标单击保存。

（2）工程文件命名保存后弹出一个工程编译选择单片机型号的窗口如图 2-19 所示，根据使用的单片机来选择相应的型号，如 Atmel 公司的 AT89C51，注意这里选择单片机型号只是软件上的编译，与实际使用的型号没有直接的关联，使用的时候可以选择其他型号的单片机，只要是 51 系列有 40 个引脚都可以。型号选择正确后鼠标单击 OK 弹出一个代码添加界面，如图 2-20 所示，在进入 C 语言编程之前执行一段汇编代码，并添加到工程中，鼠标单击 Y 添加成功，工程文件建立完成界面如图 2-21 所示。

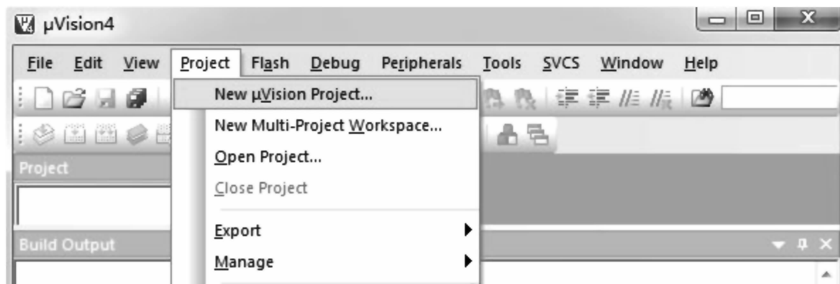



图 2-17 创建工程菜单

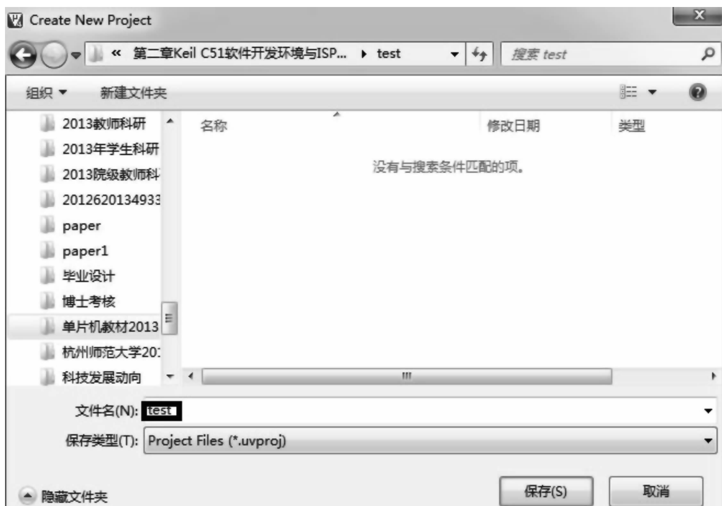



图 2-18 创建一个新的工程

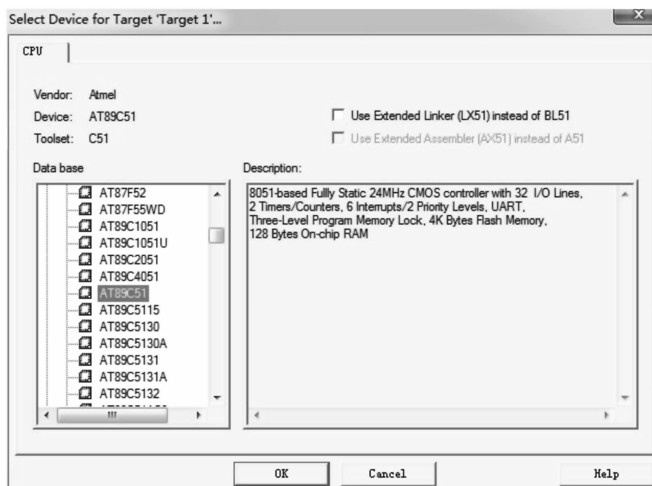



图 2-19　选择单片机型号窗口

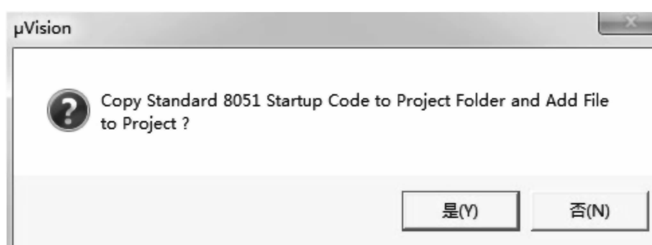



图 2-20　代码添加界面

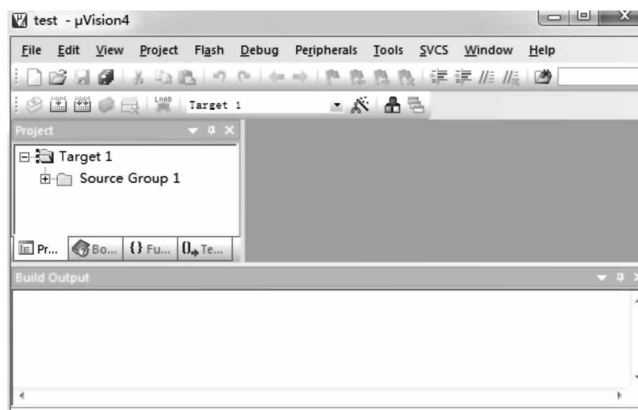



图 2-21　工程创建成功后的工作界面

（3）在创建工程成功的界面下进行 C 语言编程，执行 File 菜单中的 New 选项如图 2-22 所示，新建一个程序编译文件，新建的文件界面如图 2-23 所示，可以在 text2 区域编写 C 语言程序。

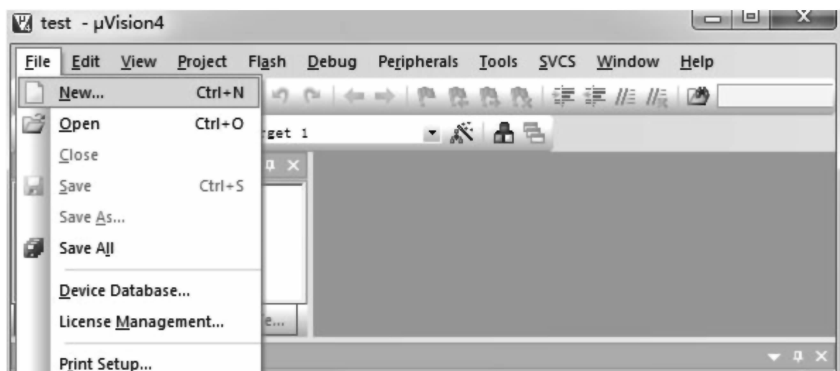



图 2-22　执行新建程序编辑指令

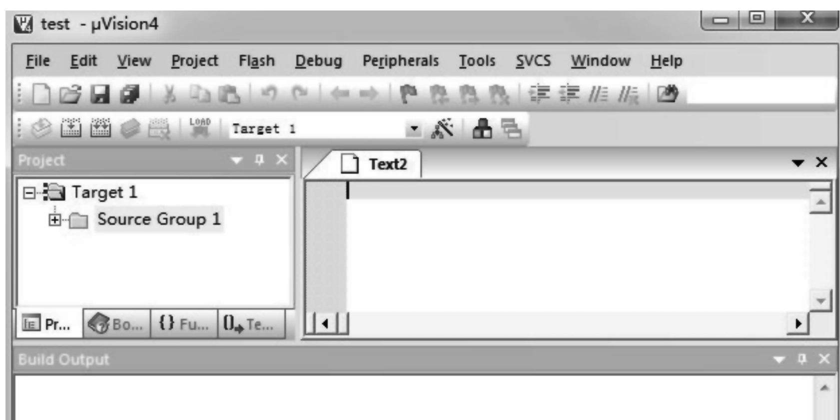



图 2-23　新建完成的程序编辑界面 text2

（4）编写一段流水灯移位的 C 语言程序在 text2 区域内，程序如下：

```
#include <reg51.h>
#include  <intrins.h>
void delayms(unsigned char ms)          // 延时子程序
{
    unsigned char i;
    while(ms--)
    {
        for(i=0; i < 120; i++);
    }
}
```

```
main()
{
    unsigned char LED;
    LED = 0xfe;
    P3 = LED;
    while(1)
    {
        delayms(250);
        LED = _crol_(LED,1);          //循环右移 1 位,点亮下一个 LED
        P3 = LED;
    }
}
```

将编辑好的程序书写到 text2 区域内，鼠标单击保存按钮。在输入程序时，Keil C51 会自动识别关键字，并以不同的颜色提示用户注意，减少用户编写程序出现的错误，有利于提高编程效率。屏幕如图 2-24 所示，在文件名栏右侧的编辑框中，输入使用的文件名如 test，同时必须输入正确的扩展名。注意使用 C 语言编写程序，则扩展名为（.c）；若采用汇编语言编写程序，则扩展名为（.asm）。本书只是讨论用 C 语言编写程序，汇编不做讨论，单击保存按钮。




图 2-24　保存编辑的 C 程序文件

（5）返回新建工程界面，单击 Target1 前面的“+”号，在出现的 source group1 上鼠标右键单击，弹出添加 test 编辑好的 C 语言文件加入到 source group1 中如图 2-25 所示。

出现图 2-26 添加文件界面，鼠标选中 test.c 文件单击 Add，会看到在 source group1 中有 test.c 文件加入成功如图 2-27 所示。

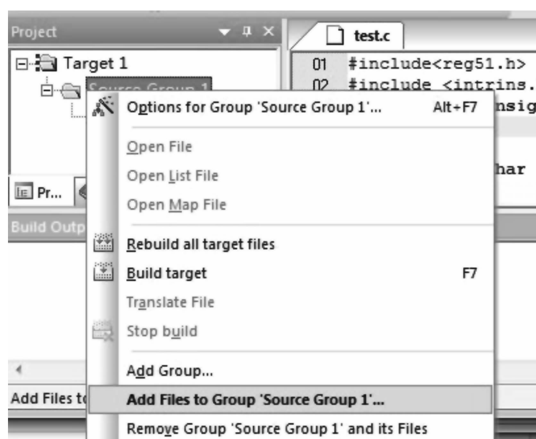



图 2-25　将程序文件加入到 source group1 中

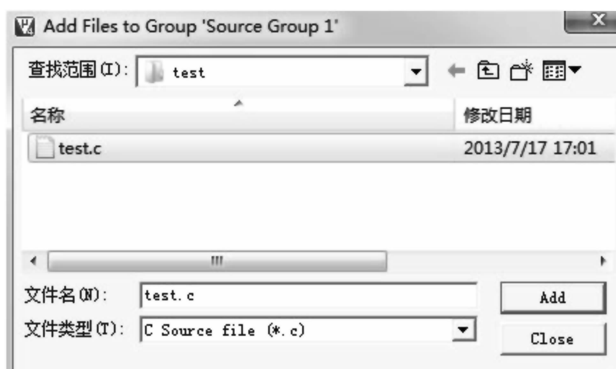



图 2-26　添加文件界面

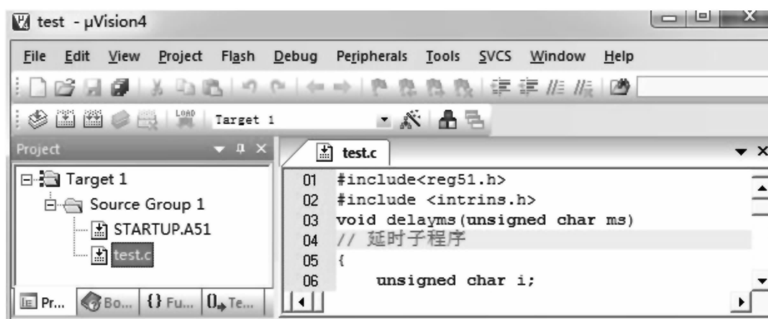



图 2-27　文件加入成功界面

（6）以上工程和编写的程序创建完成，接下来对编写的程序进行编译来检验程序的正确性。单击 project 菜单中的 built target 选项（快捷键 F7）如图 2-28 所示，进行程序编译。输出编译的结果如图 2-29 所示。

若编写的程序无错误，则输出结果提示“0 Error（s），0 Warning（s）”，可以将程序利用 2.4 节的知识下载到单片机中实现单片机对设计的硬件进行控制。

若编写的程序有错误，则输出结果会有错误提示如“Target not created”目标无法创建，需要对程序进行一步步调试直到找到错误，程序修改正确为止。

（7）调试是为了检查程序中看不见的错误，程序调试包括单步执行和全速执行。单步执行是每次执行一行程序，执行完该行程序即停止，等待命令执行下一行程序，此时可以观察该行程序执行完后得到的结果是否与我们设计的结果一致，以便发现程序存在的问题；全速执行指的是执行一段程序时，中间不间断，执行速度快可以看到程序最终的结果，但是若程序本身有错，全速执行很难发现错误的地方；在调试程序时可以将单步执行和全速执行交叉使用，确保调试的程序的准确性。

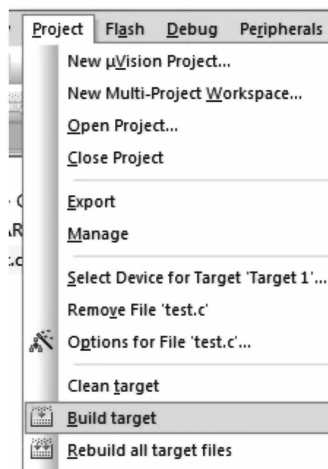



图 2-28　对编辑的程序执行编译

在对设计的程序进行调试时，单击 debug 菜单选中 start/stop debug session（快捷键 ctrl + F5），则进入到 Keil C51 程序调试界面如图 2-30 所示。

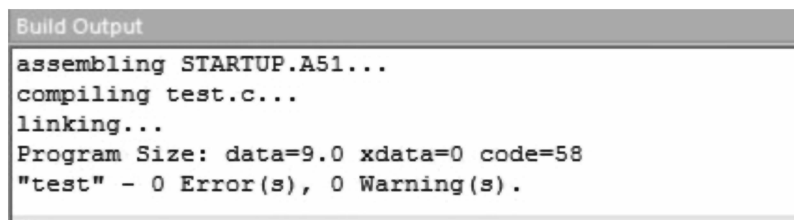



图 2-29　程序编译后的输出结果

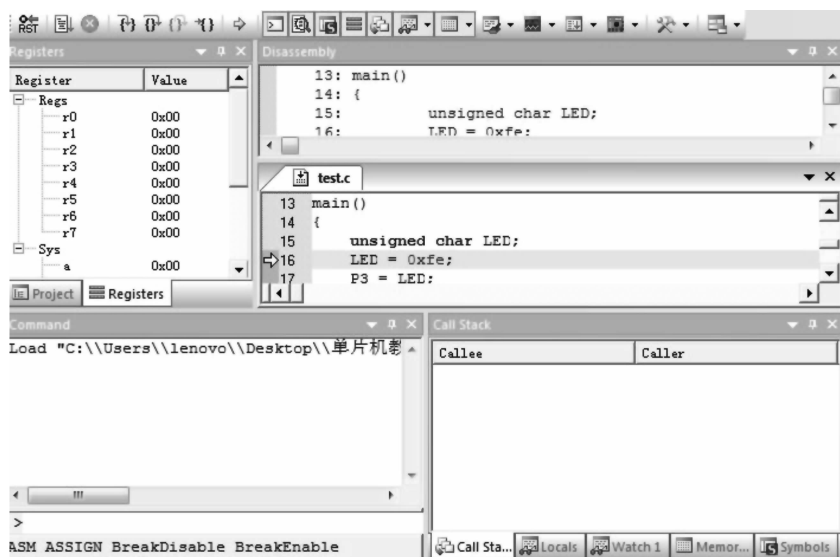



图 2-30　调试工作界面

全速执行时，单击 debug 菜单中的 run 选项，对编辑的程序进行调试运行，再单击 debug 菜单中的 stop 选项，则程序编译调试结束如图 2-31 所示。查看程序运行结果可以通过观察窗口（watch & call stack window）来观察变量值 P3 值的变化，通过执行 view 菜单中的 watch windows 中的 watch 1 如图 2-32 所示，查看 P3 调试结果如图 2-33 所示。

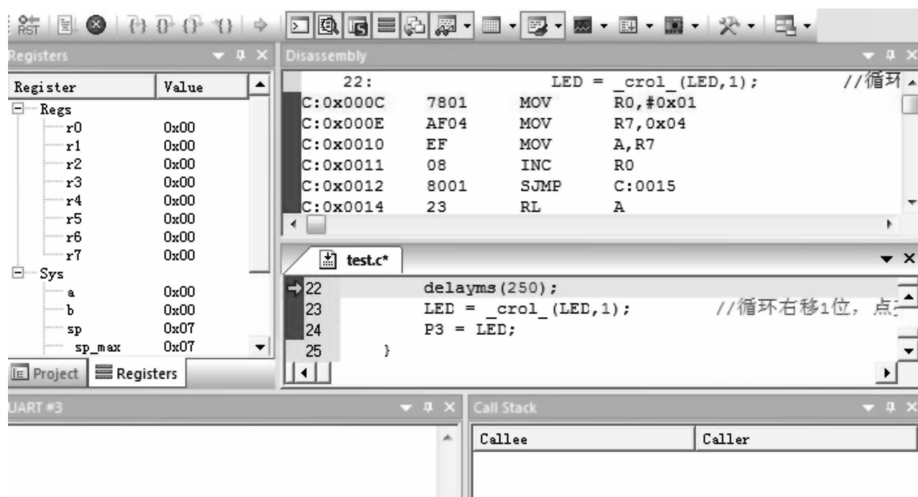



图 2-31　全速调试执行结果

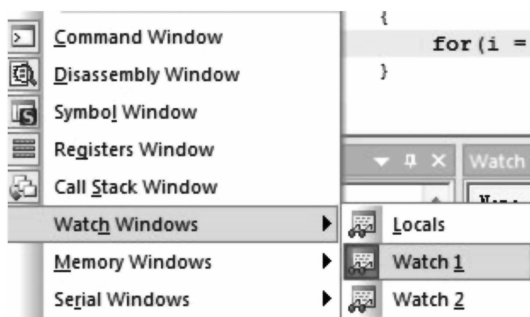



图 2-32　观察窗口执行菜单

单步执行时，灵活地使用调试快捷选项，进行单步调试，单步调试结果如图 2-34 所示，大大提高了调试程序的效率。

：单步执行命令。

：单步执行遇到循环子程序时，选择过程单步命令不会进入循环子程序内部。

：当次数很多的循环子程序中，选择单步执行到函数外命令。

：当次数很多的循环子程序中，运行到光标所在行命令跳出循环子程序。

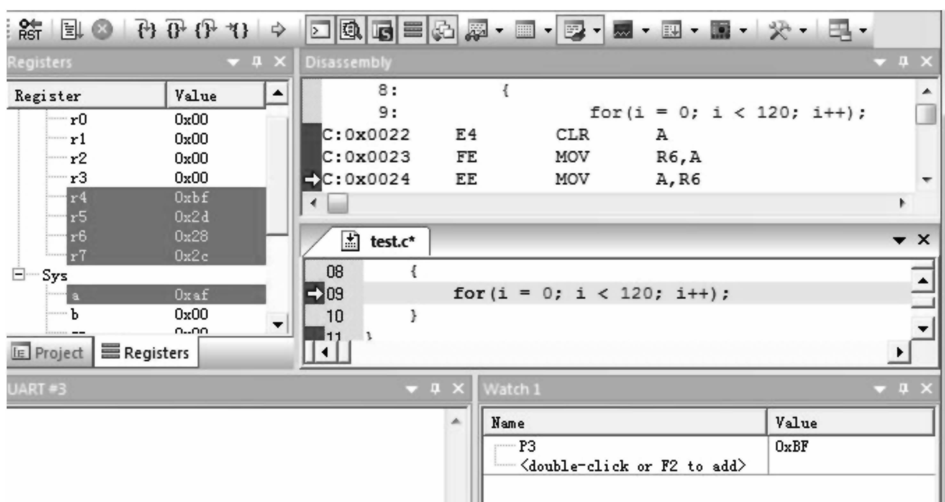



图 2-33　观察窗口查看 P3 值变化

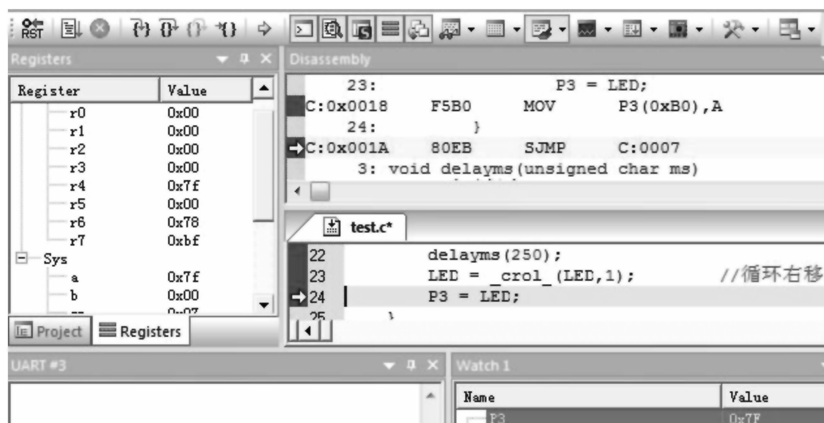



图 2-34　单步调试结果

（8）另外，在调试程序时，有时候程序需满足一些特定的条件才能执行，如程序中某些变量达到一定的值，按键被按下，串口接收到数据，定时或者中断发生等情况，单步调试很难实现，此时需要设置断点才能调试。所谓的断点调试指的是在一段程序中设置断点，一旦调试运行到该断点则程序停止，可以观察设置断点处的变量值找出问题。将光标定位于需要设置断点的程序中，通过执行 debug 菜单中的 inster/remove breakpoint 设置和删除断点如图 2-35 所示，可以设置断点亦可以删除该断点，设置好的断点如图 2-36 所示。对设定好的断点调试运行，调试结果如图 2-37 所示。

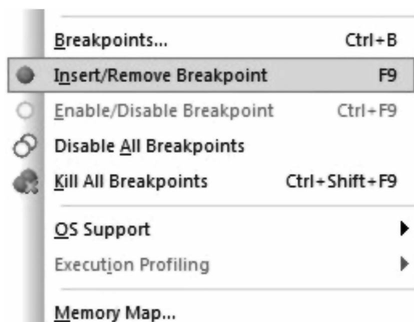



图 2-35　设置和删除断点操作

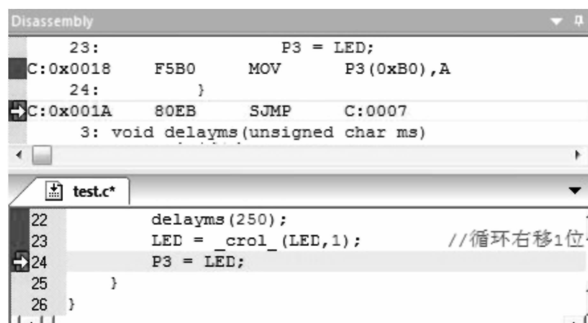



图 2-36 断点设置成功界面

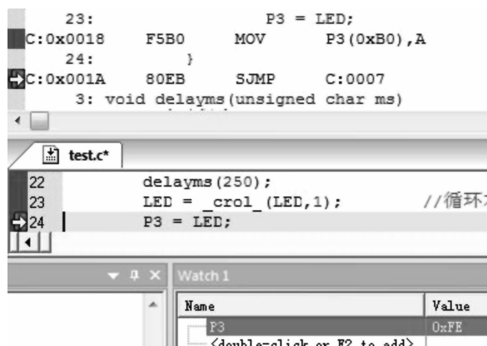



图 2-37 断点设置成功后调试结果图

（9）以上为 Keil C51 完整的工程应用过程和软件开发过程，若程序调试无错误，需要将调试正确的程序下载到单片机芯片中，供单片机驱动设计的电路正常工作。通过单击 project 菜单中的 options for target“target 1”，弹出 options for target 选项如图 2-38 所示。

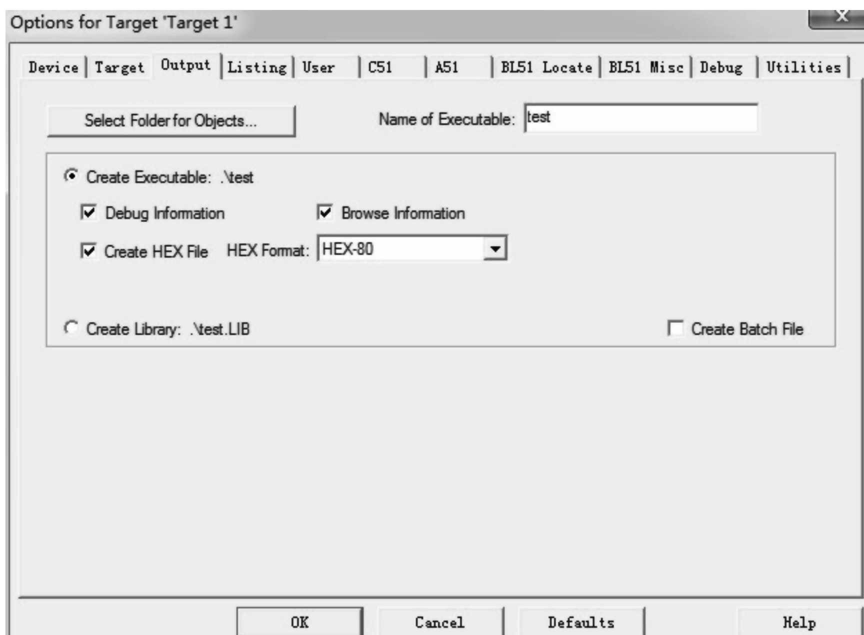



图 2-38 options for target 选项图

在图 2-38 中单击 output，选中 create HEX File 选项，使得程序编译后产生 HEX 代码，重新对 test. c 文件进行 built target 二次编译，在创建工程路径文件夹下生成一个扩展名为 . hex 文件，供下载软件将生成的 . hex 文件下载到单片机中。

## 2.4 ISP 在线下载操作

51 系列单片机中的 STC89 系列和 AT89 系列是兼容的，引脚定义和内部资源是一样的，所以烧写程序的方法也是一致，STC89 系列可以直接用 USB 或者串口下载，AT89 系列单片

机采用 ISP 下载线或者编程器来下载程序，一种方式是并口或者串口 ISP，另一种方式是 USB 转换为 ISP 下载。

本节主要介绍下载软件 STC-ISP 在 Windows 7 系统中的应用，网络下载 STC-ISP 安装软件，在软件包中找到 STC_ISP_V480.exe 可执行文件，鼠标右击在弹出选项中找到属性选项如图 2-39 所示。在弹出的属性对话框中找到兼容性选项，选中“以兼容模式运行这个程序”的同时选中“以管理员身份运行此程序”，如图 2-40 所示，鼠标单击确定。




图 2-39　STC-ISP 属性菜单操作

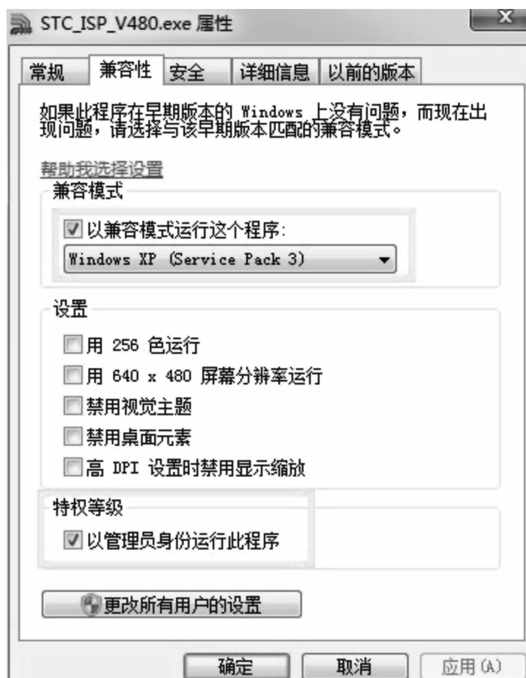



图 2-40　STC-ISP 属性对话框

设置完成后运行 STC_ISP_V480.exe，STC-ISP 可以正常进行程序烧写工作，如图 2-41 所示。




图 2-41　STC-ISP 程序烧写界面

目前便携式计算机很多无串口，程序烧写时，可以采用 USB 转 ISP 下载，若是台式计算机则可以用串口线直接进行 ISP 下载，本节采用 USB 转 ISP 下载方式，在烧写程序前安装 USB 驱动程序 PL2303 USB 转串口驱动，安装成功后鼠标右击桌面“计算机”，选择“属性”找到“设备管理器窗口”如图 2-42 所示，可以看到选择串口 COM7，同时将烧写的单片机 STC89C52RC 插入到烧写器中，确保硬件连接正确，此时烧写器无上电。




图 2-42 串口安装成功

打开 STC-ISP 软件如图 2-43 中，在 MCU 类型中选择所用的单片机型号 STC89C52RC，在 COM 串口选择中选中找到的 COM7 串口，鼠标单击打开程序，选择要烧写的程序文件如 test. hex，如图 2-44 所示，注意这里用到的烧写文件为 2. 3 节程序编译后产生 HEX 代码文件。




图 2-43 程序烧写 STC-ISP 参数配置界面

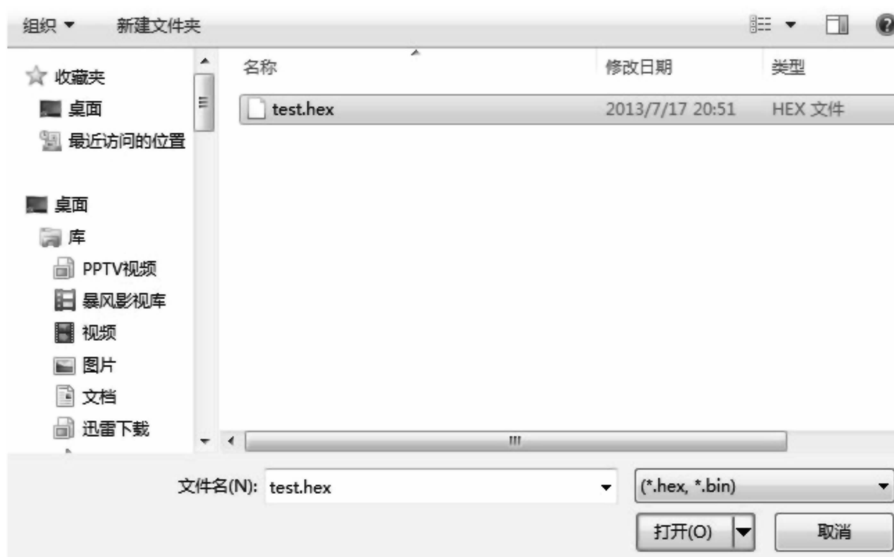



图 2-44 选择烧写 HEX 代码文件

全部准备工作完成后开始程序烧写，在图 2-45 所示界面中选中 download 下载，下载进程如图 2-45 所示，因为 STC-ISP 程序下载需要冷启动，冷启动是让单片机工作从没有电到有电的过程，所以选中 download 下载按钮后，若确保程序正常下载，需要给单片机上电，

此时给烧写器中的单片机上电，开始把编译正确的 test. hex 代码文件通过烧写器以及 STC-ISP 下载软件烧写到单片机中，烧写程序成功界面如图 2-46 所示，界面中已经显示程序下载成功。将烧写好程序的单片机插入到单片机开发板中，上电就可以看到单片机 P3 口控制的 8 个 LED 等循环点亮，呈现流水灯工作状态，程序调试成功。

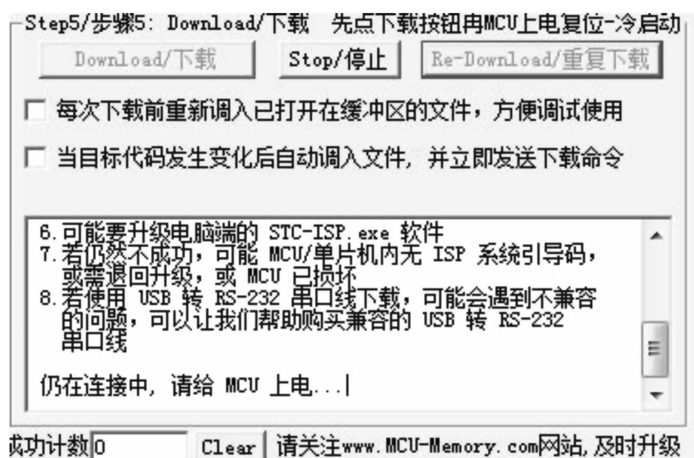



图 2-45　程序下载进程（1）

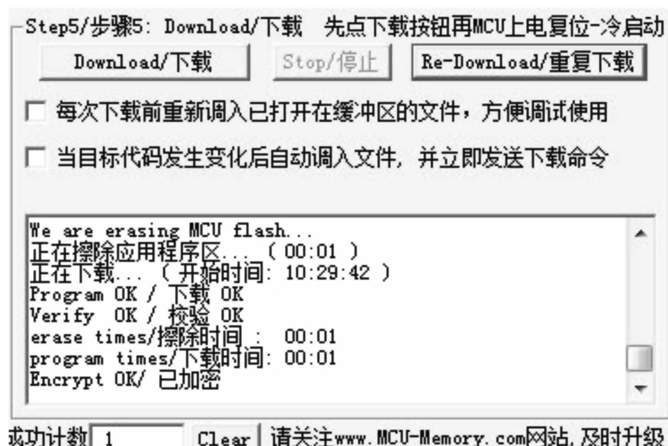



图 2-46　程序下载进程（2）

## 2.5　本章小结

在进行 51 单片机程序设计时，需要对程序进行编译和调试确保设计程序的正确性。因此本章详细地介绍了 51 单片机程序设计时使用的 Keil C51 编译和调试环境，包括 Keil C51 程序的安装、卸载、Keil C51 界面操作、单片机程序编辑、工程文件建立和应用、程序后期编译和调试，以及如何将调试正确的程序烧写到单片机中。通过本章的学习，读者可以对 51 单片机程序的编译、调试和下载工作快速上手，能够快速地学习 Keil C51 编译器的操作以及如何用 Keil 开发应用程序。

# 第 3 章　51 单片机硬件系统及体系结构

单片机（single chip microcomputer）是一种集成芯片，主要功能是在单硅晶片上集成微型计算机的主要功能。简单地讲单片机是一个微型计算机系统，内部集成了输入/输出口、中央处理器（CPU）、随机数据存储器（RAM）、只读程序存储器（ROM）、定时器/计数器、以及串行通信等主要功能器件。目前，常用的是 8051 系列，不同 51 系列单片机的内部结构基本相同，本章以 8051 单片机为例，从初学者入门的角度针对性地介绍 51 单片机硬件系统和体系结构。

## 3.1　单片机基本结构与引脚功能

### 3.1.1　单片机基本结构

本书所讲的 51 系列单片机指的是 MCS-51 系列单片机，可能不同厂家生产的 51 系列单片机从表面到内部资源不完全一样，但是芯片的内部结构是一致的，都采用了 8051 内核。51 单片机内部基本模块包括中央处理器（CPU）、存储器（数据存储器 RAM 和程序存储器 ROM）、并行 I/O 口、串行通信口、定时/计数器、中断系统等功能器件，内部模块之间通信由总线连接，51 单片机内部结构框图如图 3-1 所示。

8051 单片机的内部结构由运算器、控制器、存储器（ROM 及 RAM）、数据总线和 I/O 接口组成。中央处理器 CPU 包括运算器、控制器以及若干寄存器等部件。

运算器电路包括 ALU（算术逻辑单元）、ACC（累加器）、B 寄存器、状态寄存器 PSW、暂存器 TMP1 和暂存器 TMP2 等部件，运算器的功能是进行算术运算、逻辑运算和位运算。

**1. 累加器 A**

累加器为 8 位寄存器，是程序中最常用的专用寄存器，在指令系统中累加器的助记符为 A。作用：作为暂存寄存器，用于提供操作数和存放运算结果。直接与内部总线相连，一般信息传递和交换都要经过 ACC。

**2. 寄存器 B**

B 寄存器为 8 位寄存器，主要用于乘除指令中。乘法指令的两个操作数分别取自累加器 A 和寄存器 B，其中 B 为乘数，乘法结果的高 8 位存放于寄存器 B 中。除法指令中，被除数取自 A，除数取自 B，除法的结果商数存放于 A，余数存放于 B 中。在其他指令中，B 寄存器也可作为一般的数据单元来使用。

**3. 程序状态字 PSW**

程序状态字是一个 8 位寄存器，它包含程序的状态信息。在状态字中，有些位状态是根据指令执行结果，由硬件自动完成设置的，而有些状态位则必须通过软件方法设定。PSW 中的每个状态位都可由软件读出，PSW 的各位定义如下：

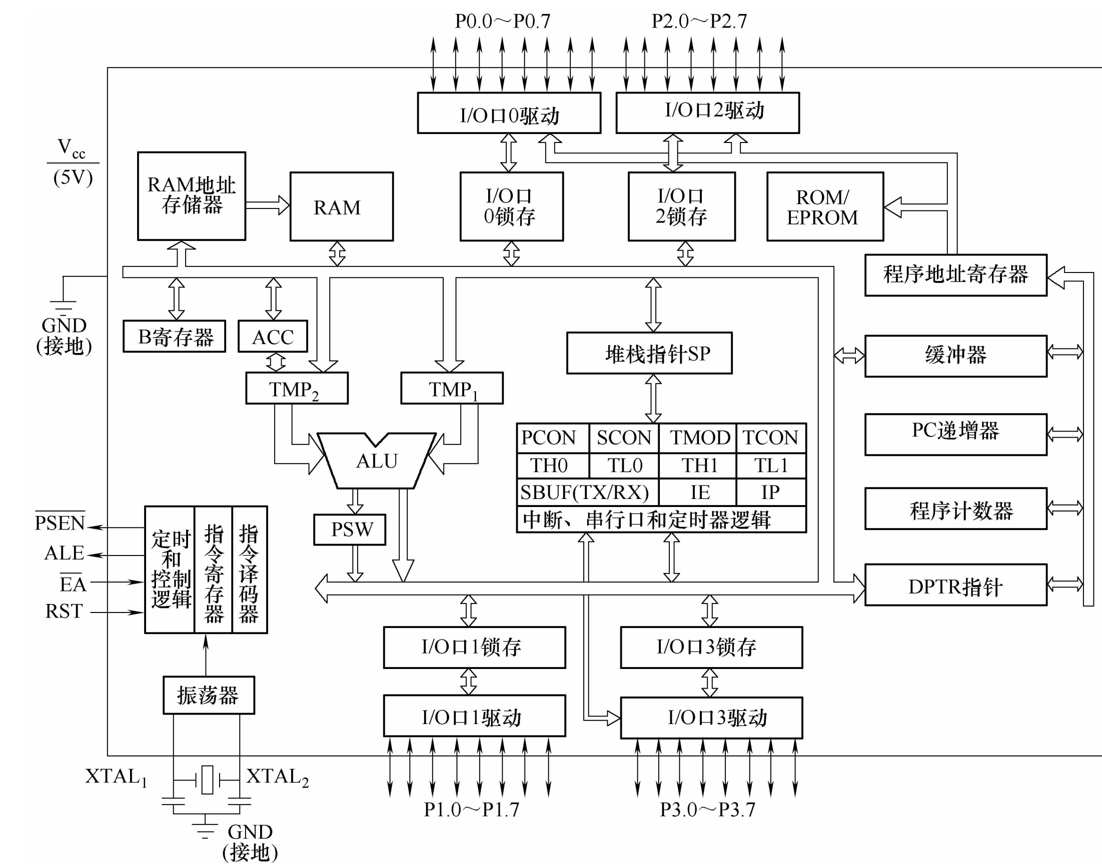



图 3-1　51 单片机内部结构框图

| 位 序 | PSW. 7 | PSW. 6 | PSW. 5 | PSW. 4 | PSW. 3 | PSW. 2 | PSW. 1 | PSW. 0 |
|---|---|---|---|---|---|---|---|---|
| 位标志 | CY | AC | F0 | RS1 | RS0 | OV | / | P |

（1）CY：进位标志位

在执行某些算术和逻辑指令时，可以被硬件或软件置位或清零。在算术运算中它可作为进位标志，在位运算中，它作累加器使用，在位传送、位与和位或等位操作中，都要使用进位标志位。

（2）AC：辅助进位标志

进行加法或减法操作时，当发生低四位向高四位进位或借位时，AC 由硬件置位，否则 AC 位被置“0”。在进行十进制调整指令时，将借助 AC 状态进行判断。

（3）F0：用户标志位

该位为用户定义的状态标记，用户根据需要用软件对其置位或清零，也可以用软件测试 F0 来控制程序的跳转。

（4）RS1 和 RS0：寄存器区选择控制位

该两位通过软件置“0”或“1”来选择当前工作寄存器区。

（5）OV：溢出标志位

当执行算术指令时，由硬件置位或清零来指示溢出状态。在带符号的加减运算中，OV =1 表示加减运算结果超出了累加器 A 所能表示的符号数有效范围（－128 ~127），即运算结果是错误的，反之，OV =0，即无溢出产生。

（6）P：奇偶标志位

每个指令周期由硬件来置位或清零，用以表示累加器 A 中 1 的个数的奇偶性，若累加器中 1 的个数为奇数则 P =1，否则 P =0。

51 单片机各部分功能介绍如下：

1）中央处理器（CPU） 单片机的核心，完成运算和控制功能，MCS-51 的 CPU 能处理 8 位二进制数或代码。

2）内部数据存储器（内部 RAM） 8051 芯片中有 128 个 RAM 单元，用于存放可读写的数据，简称内部 RAM。

3）内部程序存储器（内部 ROM） 8051 芯片内部有 8KB 掩膜 ROM，用于存放程序或表格，因此称为程序存储器，简称内部 ROM。

4）定时器/计数器 8051 共有两个可编程的 16 位定时器/计数器，以实现定时或计数功能。

5）并行 I/O 口 MCS-51 单片机共有 4 个可编程的 8 位 I/O 口（P0，P1，P2，P3），以实现数据的并行输入/输出。

6）串行口 MCS-51 单片机有一个全双工的串行 I/O 口，以实现单片机和其他设备之间的串行数据传送。

7）中断控制系统 8051 单片机拥有 5 个中断源，两个中断优先级的中断控制系统，以满足控制应用的需要。

8）时钟电路 由振荡电路得到一个基本的振荡信号，然后对振荡信号进行分频，从而得到相应的时钟。单片机的振荡电路有两种：内部振荡（电路设计时使用比较多）和外部振荡。

内部振荡如图 3-2 所示：MCS-51 芯片的内部有一个用于构成振荡器的高增益反相放大器，引脚 XTAL1 和 XTAL2 分别为放大器的输入端和输出端，将其余外接的晶体振荡器和微调电容相连，构成内部自激振荡器产生振荡时钟脉冲，电容 C1 和 C2 电容容量选择范围为 5 ~30pF，系统允许的晶振频率为 1.2 ~12MHz（此种情况电路设计时使用比较多）。

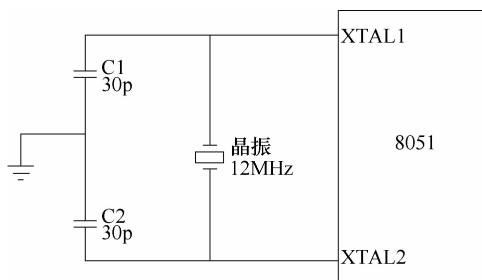


图 3-2 单片机内部振荡

外部振荡：利用外部已有的时钟信号引入到单片机内部。对于 HMOS 8051 单片机外部时钟 XTAL2 用来输入外部时钟信号，XTAL1 接地如图 3-3 所示；对于 CHMOS 80C51 单片机必须用 XTAL1 输入外部时钟信号，XTAL2 悬空如图 3-4 所示。（注意：8051 与 80C51 完全兼容，差别主要在芯片制造工艺上，80C51 的制造工艺是在 8051 基础上进行了改进，8051 采用 HMOS 工艺，高速度、高密度；80C51 采用 CHMOS 工艺，高速度、高密度和低功耗）。

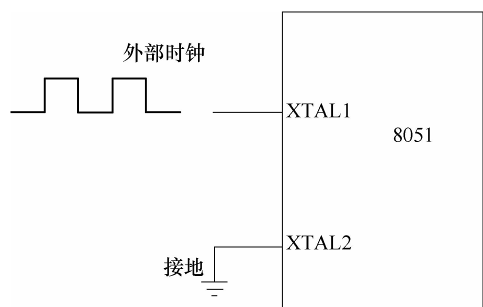



图 3-3　HMOS8051 单片机外部时钟

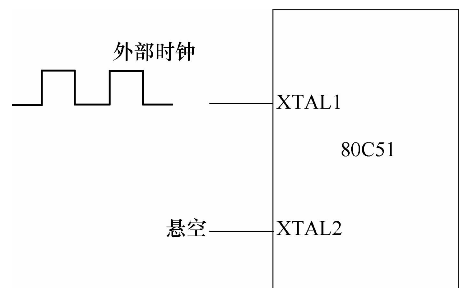



图 3-4　CHMOS80C51 单片机

9）时序定时 51 单片机共有 4 个，分别为振荡周期、状态周期、机器周期和指令周期。

①振荡周期：为单片机提供时钟信号的振荡源周期，振荡信号若采用内部振荡则周期为石英晶振频率的倒数。

②状态周期：振荡源信号经二分频后形成的时钟脉冲信号，振荡信号若采用内部振荡则状态周期为石英晶振频率/2 的倒数。

③机器周期：完成一个基本操作所需的时间。

④指令周期：CPU 执行一条指令所需要的时间，一般一个指令周期含有 1 ~4 个机器周期。

## 3.1.2　单片机引脚功能

51 单片机一般采用 40 引脚双列直插（DIP）封装，图 3-5 为单片机引脚排列图。其中 40 个引脚可以分为 4 类：电源、控制、I/O 端口和时钟引脚。

单片机的引脚学习非常重要，通过引脚的功能定义可以对单片机进行硬件电路设计，尤其对于初学者知道每个引脚的功能方便对单片机开发应用。这里对单片机 40 个引脚从 4 大类分别详细介绍：

**1. 电源引脚**

①40 引脚 VCC：供电电压 5V。

②20 引脚 GND：接地。

**2. I/O 端口引脚**

51 单片机共有 4 个 8 位并行 I/O 端口，共 32 个可编程 I/O 引脚。每个端口介绍如下：

①P0 口：32 引脚（P0.7）~39 引脚（P0.0）可作为普通的输入/输出端口用。访问外部存储器时，分时传送低 8 位地址信号和低 8 位数据信号。

②P1 口：8 引脚（P1.7）~1 引脚（P1.0）

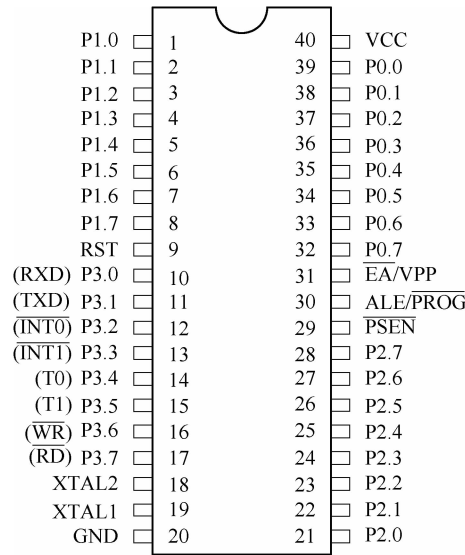



图 3-5　单片机引脚排列图

只有通用 I/O 接口一种功能，作为普通的输入/输出端口用。

③P2 口：28 引脚（P2.7）~21 引脚（P2.0）可作为普通的输入/输出端口用。访问外部存储器时传送高 8 位地址信号。

④P3：17 引脚（P3.7）~10 引脚（P3.0）可作为普通的输入/输出端口用。但大部分情况下用于第二功能，见表 3-1。

**表 3-1　P3 口的第二功能**

| P3 口引脚 | 功能描述 | P3 口引脚 | 功能描述 |
|---|---|---|---|
| P3.0（10 引脚） | RXD 串行输入口 | P3.4（14 引脚） | T0 定时器/计数器的外部输入 |
| P3.1（11 引脚） | TXD 串行输出口 | P3.5（15 引脚） | T1 定时器/计数器的外部输入 |
| P3.2（12 引脚） | $\overline{\text{INT0}}$外部中断 0 输入 | P3.6（16 引脚） | $\overline{\text{WR}}$片外数据存储器写选通控制输出 |
| P3.3（13 引脚） | $\overline{\text{INT1}}$外部中断 1 输入 | P3.7（17 引脚） | $\overline{\text{RD}}$片外数据存储器读选通控制输出 |

**3. 时钟引脚**

时钟引脚 XTAL1 和 XTAL2，电路设计时多采用内部振荡方式产生时钟脉冲。详细介绍参见 3.1.1 节。

19 引脚 XTAL1：接内部时钟工作电路的输入。

18 引脚 XTAL0：接内部时钟工作电路的输出。

**4. 控制引脚**

51 单片机的控制线共有 4 根，分别为 RST、$\overline{\text{PSEN}}$、ALE/$\overline{\text{PROG}}$和$\overline{\text{EA}}$/VPP。其中 3 根是复用线为$\overline{\text{PSEN}}$、ALE/$\overline{\text{PROG}}$和$\overline{\text{EA}}$/VPP 具有两种功能。

1）复位引脚 RST：9 引脚 RST 为单片机上电复位输入端，只要在该引脚上连续保持两个机器周期以上的高电平，单片机就可以实现复位操作。复位电路分为自动上电复位和手动复位，如图 3-6、图 3-7 所示。

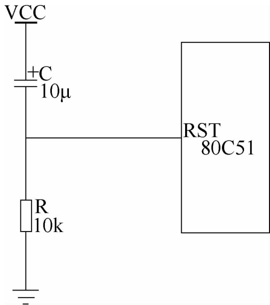



图 3-6　自动上电复位

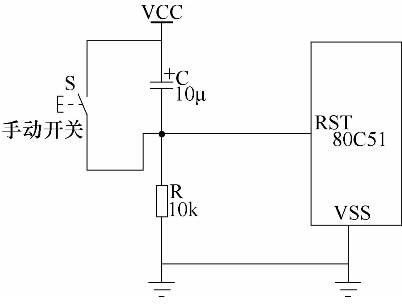



图 3-7　手动复位

2）$\overline{\text{PSEN}}$：29 引脚$\overline{\text{PSEN}}$为外部程序存储器 ROM 读选通信号。该引脚有效（低电平）时，当单片机读取外部程序存储器 ROM 的指令和数据时，每个机器周期内$\overline{\text{PSEN}}$两次有效，$\overline{\text{PSEN}}$相当于外部 ROM 芯片输出允许的选通信号。但读片内 ROM 和读片外 RAM 时，不会产生有效的$\overline{\text{PSEN}}$信号。

3）ALE/$\overline{\text{PROG}}$：30 引脚 ALE/$\overline{\text{PROG}}$为地址锁存允许/编程脉冲。访问外部存储器时，

ALE 用来锁存 P0 口送出的低 8 位地址信号。当 ALE 信号有效（高电平）时，P0 口传送的是低 8 位地址信号；当 ALE 无效（低电平）时，P0 口传送的是 8 位数据信号。

当不执行访问外部 RAM 指令时，ALE 以时钟振荡频率 1/6 的固定频率输出，ALE 信号可以作为外部芯片的时钟信号。但当 CPU 执行访问外部 RAM 时，ALE 将丢失一个 ALE 脉冲。

$\overline{PROG}$：单片机对内部 ROM 编程时的编程脉冲输入端。

4）$\overline{EA}$/VPP：31 引脚$\overline{EA}$/VPP 为访问程序存储器控制信号。51 型单片机的寻址范围为 64KB，其中 4KB 在片内，60KB 在片外，当$\overline{EA}$为高电平时，先访问片内 ROM，当程序长度超过 4KB 时将自动转向执行外部 ROM 中的程序。当$\overline{EA}$为低电平时，单片机只访问外部 ROM。

VPP：对片内 EPROM 编程，31 引脚为高电平用于施加编程电源。

## 3.2　单片机存储器

51 单片机存储器结构设计上采用哈佛结构，其程序存储器和数据存储器寻址空间完全分开，存储器包括内部数据、程序存储器、外部扩展的数据和程序存储器。51 单片机存储器地址空间分配如图 3-8 所示。

### 3.2.1　程序存储器

51 单片机芯片内部提供 4KB 字节的程序存储器，片外可用 16 位地址线扩展到 64KB 的 ROM 空间。CPU 可以通过单片机 31 引脚$\overline{EA}$/VPP 选择由内部的程序区启动或者由外部的程序区启动，如图 3-8 所示：$\overline{EA}$ = 0 外部扩展 ROM 到 64KB 的空间，地址从 0000H 开始编码；$\overline{EA}$ = 1，执行芯片内部程序，一旦程序存储器空间超过 4KB 字节时，CPU 自动选择外部 ROM 执行程序。程序存储器有部分重叠，在某些单元被保留做特定程序入口地址见表 3-2。

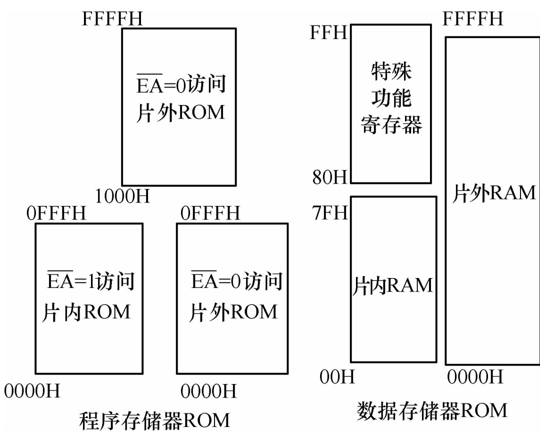



图 3-8　51 单片机存储器映像图

**表 3-2　特定程序入口地址表**

| 描　述 | 地　址 | 描　述 | 地　址 |
|---|---|---|---|
| 单片机复位单元 | 0000H | 外部中断 1 入口地址 | 0013H |
| 外部中断 0 入口地址 | 0003H | 定时/计数器 T1 溢出中断入口地址 | 001BH |
| 定时/计数器 T0 溢出中断入口地址 | 000BH | 串行通信口中断入口地址 | 0023H |

### 3.2.2　数据存储器

51 单片机数据存储器用于存放运算中间结果、数据暂存和数据缓冲、标志位等。数据存储器包括片内 RAM 和片外 RAM 两种，采用不同的方法访问，不存在重叠的现象，从图

3-8 所示片外 RAM 空间为 64KB，地址范围从 0000H ~ FFFFH，对 51 单片机低 128B 地址空间，00H ~ 7FH 为片内 RAM 作为处理问题的数据缓冲器；其地址分配如图 3-9 所示；51 单片机有 32 个工作寄存器，00H ~ 1FH 为四组工作寄存器区，每组有 8 个工作寄存器见表 3-3。CPU 在复位后默认选择表 3-3 中第 0 组工作寄存器。

工作寄存器从 20H ~ 2FH 为位寻址区，位寻址区的每一位可以由程序直接进行位处理，图 3-9 中给出了自己地址和位地址对应关系，竖向字节地址从 20H ~ 2FH，横向代表 8 位位地址分别为 D7 ~ D0，如图 3-9 中 7FH 为字节地址 2FH，位地址 D7 位。

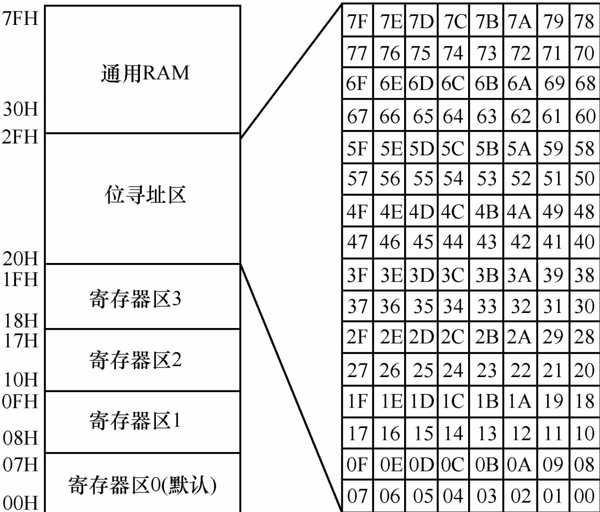



图 3-9　片内 RAM 地址分配图

**表 3-3　工作寄存器地址列表**

| 选择工作寄存器组 | RS1 | RS0 | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 00H | 01H | 02H | 03H | 04H | 05H | 06H | 07H |
| 1 | 0 | 1 | 08H | 09H | 0AH | 0BH | 0CH | 0DH | 0EH | 0FH |
| 2 | 1 | 0 | 10H | 11H | 12H | 13H | 14H | 15H | 16H | 17H |
| 3 | 1 | 1 | 18H | 19H | 1AH | 1BH | 1CH | 1DH | 1EH | 1FH |

高 128B 地址空间，80H ~ FFH 为特殊功能寄存器空间。其地址范围为 80H ~ FFH，特殊功能寄存器地址空间分配如图 3-10 所示，详细描述见表 3-4。

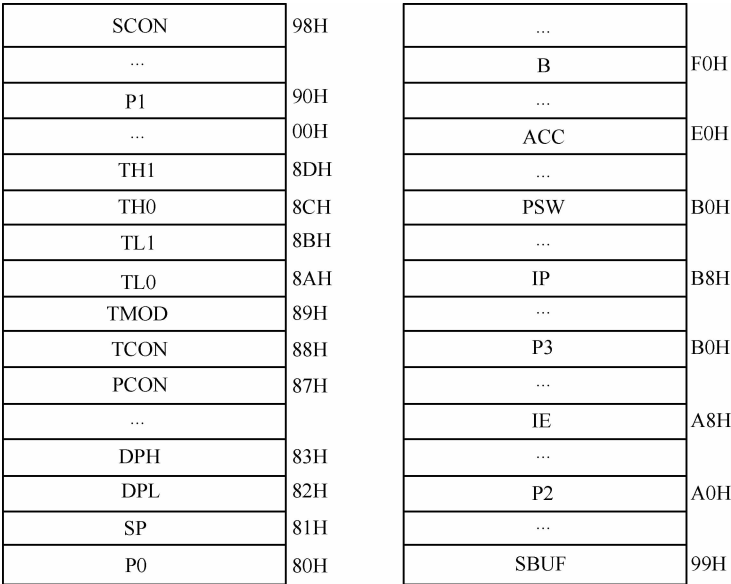



图 3-10　特殊功能寄存器地址分配图

表 3-4　特殊功能寄存器说明

| 特殊功能寄存器描述 | 符号 | 特殊功能寄存器描述 | 符号 |
| --- | --- | --- | --- |
| 累加器 A | ACC | IE | 中断允许控制 |
| 寄存器 B | B | TMOD | 定时/计数器工作方式选择 |
| 程序状态字 | PSW | TCON | 定时/计数器控制 |
| 堆栈指针 | SP | TH0 | 定时/计数器 0 高字节 |
| 数据指针 | DPTR | TL0 | 定时/计数器 0 低字节 |
| P0 | P0 端口锁存 | TH1 | 定时/计数器 1 高字节 |
| P1 | P1 端口锁存 | TL1 | 定时/计数器 1 高字节 |
| P2 | P2 端口锁存 | SCON | 串行口控制 |
| P3 | P3 端口锁存 | SBUF | 串行数据缓冲器 |
| IP | 中断优先级控制 | PCON | 电源控制 |

表 3-4 中论述的特殊功能寄存器需要在程序中处理，实际应用中与硬件操作相结合，需要开发人员按功能需求进行控制。有部分寄存器将在定时/计数器、串口、中断内容中介绍，因为本书讨论单片机的软件编程采用 C51 来设计程序内容，所有对于采用 C 语言进行单片机编程的初学者 SP、PSW、ACC、B、DPH、DPL 不再是非常重要的特殊功能寄存器。所以这里主要讨论 P0、P1、P2、P3 端口锁存寄存器，P0 ~ P4 为 4 个 8 位特殊功能寄存器，分别是 4 个并行 I/O 端口的锁存器，每个端口有字节地址也有位地址，I/O 线独立用作输出时，数据可以锁存，用作输入时，数据可以缓冲。

通过对 P0 ~ P3 口地址的读和写改变 I/O 口状态，如下通过简单样例程序初步了解对 P0 口的读取和 P2 口的输出操作。

```
unsigned char buf;
buf = P0;                  // 读取 P0 口的数据赋值给 buf
P2 = buf;                  // 读取 buf 数据赋值给 P2 口进行输出
```

若对某个 I/O 口引脚进行操作,可以参考下面的样例程序。

```
sbit P2_4 = P2^4;          //对 P2.4 引脚修改
P2_4 =0;                   // 把 P2.4 设置成低电平
P2_4 =1;                   // 把 P2.4 设置为高电平
```

## 3.3　单片机中断系统

### 3.3.1　中断定义

单片机中只有一个 CPU，但在同一时间内可能会面临着处理很多任务的情况，在 CPU 执行过程中，外部发生了某一事件，如数据的输入和输出、定时/计数器溢出及一些更重要的中断请求 CPU 快速处理。CPU 暂时中断当前的工作，转入处理所发生的事件，完成中断服务程序后，再回到原来被中断的地方，继续执行被中止的原程序过程。这样的处理模式称为中断。实现中断功能的部件称为中断系统。

对于初学者，学习中断，第一个问题先搞清楚中断的概念，第二个问题是中断优先级问题，如中断事件一旦发生，当多个中断源同时申请中断或者CPU正在处理某中断事件时，又有另一事件申请中断，CPU必须区分哪个中断源优先级更高优先处理，这时考虑中断优先处理问题，解决方案是优先级别高的事件可以中断CPU正在处理的低优先级的中断服务程序，待完成高优先级中断事件后，再继续执行被中断的低优先级中断事件，也可称为中断嵌套。

### 3.3.2　中断系统概述

51单片机系统有5个中断源，其中两个外部中断、3个内部中断，可分为2个优先级。

1）INT0　单片机引脚P3.2，属于外部中断0。由定时/外部中断控制寄存器TCON中的IT0控制，TCON在3.3.3节中做详细介绍，当TCON中的IT0=1时是边沿触发，当IT0=0时是电平触发。

2）INT1　单片机引脚P3.3，属于外部中断1。由定时/外部中断控制寄存器TCON中的IT1控制，TCON在3.3.3节中做详细介绍，当TCON中的IT1=1时是边沿触发，当IT1=0时是电平触发。

3）TF0　定时/计数器T0溢出中断。定时功能时，计数脉冲来自芯片内部；计数功能时，计数脉冲来自芯片引脚P3.4脚。计数寄存器T0发生溢出时，发生中断请求。

4）TF1　定时/计数器T1溢出中断。定时功能时，计数脉冲来自芯片内部；计数功能时，计数脉冲来自芯片引脚P3.5脚。计数寄存器T1发生溢出时，发生中断请求。

5）RI TI　串行通信中断。单片机完成接收或者发送一组数据时，产生中断请求。

### 3.3.3　中断控制

3.3.2节中讨论的外部中断IT0、IT1和定时/计数TF0、TF1以及串行中断RI和TI，这些中断请求信号分别锁存在特殊功能寄存器TCON（定时/外部中断控制寄存器）和SCON（串行中断控制寄存器）中。其中TCON和SCON只有一部分用于中断控制，通过对中断控制寄存器各位进行配置，可实现各种中断控制功能。

（1）定时器/计数器控制寄存器TCON，字节地址88H，中断请求格式见表3-5。

**表3-5　TCON中断请求格式**

| TCON | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| 位地址 | 8FH | 8EH | 8DH | 8CH | 8BH | 8AH | 89H | 88H |

其中断有关控制位有4位，分别为IT0、IT1、IE0、IE1，功能如下：

1）IT0：外部中断0类型控制位，通过软件置“1”或清“0”，用于控制外部中断触发信号。当IT0=1时是边沿触发，单片机P3.2脚输入电平由高到低下降沿有效；当IT0=0时是电平触发。单片机P3.2脚输入低电平有效。

2）IE0：外部边沿触发中断0请求标志，当CPU采样到P3.2引脚出现有效中断请求信号时，触发IE0置“1”，当CPU响应该中断后，转向中断服务程序时由片内硬件自动清0

（只适用于边沿触发方式）。

3）IT1：外部中断 1 请求方式控制位，使用方法和 IT0 相同。

4）IE1：外部边沿触发中断 1 请求标志位，使用方法和 IE0 相同。

（2） 串行口中断控制寄存器 SCON，字节地址 98H，中断请求格式见表 3-6。

**表 3-6　SCON 中断请求格式**

| SCON | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | SM0 | SM1 | SM2 | REN | TB8 | TB8 | TI | RI |
| 位地址 | 9FH | 9EH | 9DH | 9CH | 9BH | 9AH | 99H | 98H |

其中中断有关控制位有两个，分别为 SCON 低二位锁存串行口的接收中断和发送中断标志，功能如下：

1）RI：串行口接收中断标志。若串行口接收器以方式 0 工作，每接收完一个串行数据帧，RI 被置 1；RI 被置 1 表示串行口接收器正在向 CPU 请求中断，同样 RI 标志必须由中断服务程序清“0”。

2）TI：串行口发送中断标志。串行口以方式 0 发送时，每发送完 8 位数据后由硬件使 TI 置 1。TI =1 表示串行发送器正向 CPU 发出中断请求，向串行口的数据缓冲器 SBUF 写入一个数据后就立即启动发送器继续发送。但是 CPU 响应中断请求后，转向执行中断服务程序时，TI 并不清“0”，TI 必须由用户的中断服务程序清“0”。

51 单片机有两级中断优先级，中断优先级可以由程序设定，中断源是否响应中断通过中断寄存器 IE 控制，而中断优先级则有中断优先级寄存器 IP 控制，若优先级别相同，并且同时要求中断，解决方案是以内部查询逻辑确定响应次序，具体 IE、IP 控制如下：

（1） 中断允许控制寄存器 IE 各位定义见表 3-7。

**表 3-7　IE 中断请求各位**

| IE | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | EA | — | — | ES | ET1 | EX1 | ET0 | EX0 |
| 位地址 | AFH | | | ACH | ABH | AAH | A9H | A8H |

其中与中断有关的控制位有 6 个，各位功能如下：

1）EX0：外中断 0 中断控制位。EX0 =1，允许外部中断 0 中断，EX0 =0，禁止外部中断 0 中断。

2）ET0：定时/计数器 T0 中断控制位。ET0 =1，允许定时器 T0 中断，ET0 =0，禁止定时器 T0 中断。

3）EX1：外部中断 1 中断控制位。EX1 =1，允许外部中断 1 中断，EX1 =0，禁止外部中断 1 中断。

4）ET1：定时/计数器 T1 中断控制位。ET1 =1，允许 T1 中断，ET1 =0，禁止 T1 中断。

5）ES：串行口中断控制位。ES =1 允许串行口中断，ES =0，禁止串行口中断。

6）EA：中断允许位，EA =1，每个中断源是允许还是禁止，由各自的允许位决定。EA =0，CPU 禁止所有中断。

（2） 中断优先级控制寄存器 IP 各位定义见表 3-8。

表3-8 IP中断请求各位

| IP | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | PS | PT1 | PX1 | PT0 | PX0 |
| 位地址 | | | | BCH | BBH | BAH | B9H | B8H |

其中与IP有关的中断控制位有5个，各位功能如下：

1）PX0：外部中断0优先级控制位。PX0=1，声明外中断0为高优先级中断，PX0=0定义外中断0为低优先级中断。

2）PT0：定时器0优先级控制位。PT0=1，声明定时器0为高优先级中断，PT0=0定义定时器0为低优先级中断。

3）PX1：外部中断1优先级控制位。PX1=1，声明外中断1为高优先级中断，PX1=0定义外中断1为低优先级中断。

4）PT1：定时器1优先级控制位。PT1=1，声明定时器1为高优先级中断，PT1=0定义定时器1为低优先级中断。

5）PS：串行口中断优先级控制位。PS=1，串行口中断声明为高优先级中断，PS=0，串行口定义为低优先级中断。

系统复位后IP寄存器中各位均为0，此时全部设定为低中断优先级，中断执行过程中，低中断优先级被高中断优先级中断，但是高中断优先级无法被低中断优先级中断；同级中断不能互相中断，CPU按照硬件次序决定优先权。同级优先权级别如图3-11所示。

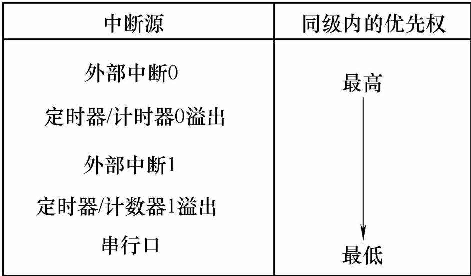



图3-11 同级优先权级别

对于IE控制位可以通过置“1”和清“0”允许或禁止中断源中断申请，IE中的EA位相当于中断允许的总开关，中断系统结构如图3-12所示。

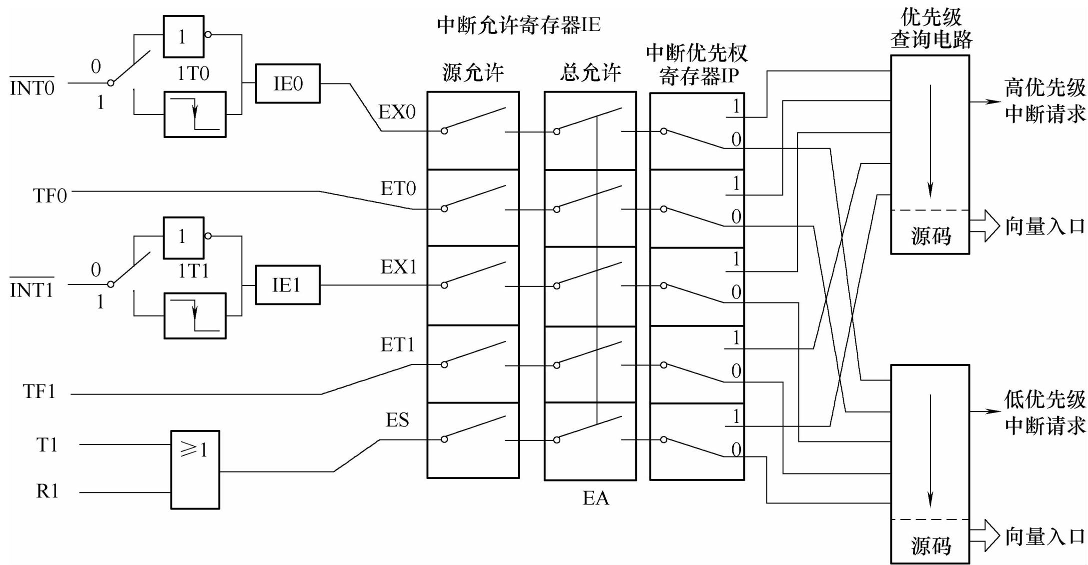



图3-12 中断结构框图

### 3.3.4　中断处理

CPU 对中断处理过程包括中断初始化、中断响应、保护现场、中断服务程序处理、中断撤销和中断返回。

**1. 中断初始化**

中断初始化设置，对中断的允许位和中断优先级设置，包括 TCON、SCON、IE、IP 与中断有关的功能位置进行设置，如 3.3.3 节所述。

**2. 中断响应**

单片机一旦响应中断，首先设置响应的优先级，通过执行硬件程序调用把断点地址压入堆栈，并与各中断源对应的中断服务程序首地址送到程序计数器 PC，同时清除中断请求标志（TI 和 RI 除外），从而控制程序转移到中断服务程序，中断响应过程由中断系统自动完成。

**3. 保护现场**

CPU 在执行任务过程中由于中断响应执行中断事件，为了在执行完中断服务程序后，回头执行原先程序时，知道程序原来在何处打断，各有关寄存器的内容如何，必须在转入执行中断服务程序前，将当前状态内容进行备份以便保护好现场。一旦缺乏现场保护和恢复现场，程序运行混乱，单片机无法正常工作。所以在进入中断服务子程序后，一般都要保护现场，再执行中断服务程序，返回主程序前再恢复现场，用户在编制中断处理程序时必须要考虑上述问题。

**4. 中断服务程序处理**

中断服务程序一般以子程序的形式出现，所有的中断都要转去执行中断服务程序，进行中断服务。C51 编辑器支持在 C 语言程序中直接编写 51 单片机的中断服务程序，C51 编译器对函数的定义进行了扩展，无需考虑堆栈出栈保护问题，增加了一个扩展关键字 interrupt。关键字 interrupt 是函数定义时的一个选项，加上此选项可以对函数定义为中断服务函数。定义中断服务函数的格式如下：

函数类型　函数名(形式参数)[interrupt　n]　[using　n]

关键字 interrupt 中的 n 是中断号，n 的取值为 0 ~ 31 单片机芯片型号不同 n 不同。

在 C 语言中定义中断服务函数：

void intersvr0（void）　interrupt 0 using 1　//定义外部中断 0，使用第一组寄存器。

**5. 中断请求撤销**

CPU 响应中断请求后，执行中断服务子程序，一旦中断任务执行完毕，必须清除中断请求标志，否则中断响应返回后将再次进入该中断，从而进入死循环。

中断撤销方式有 4 种方式：

1）对定时/计数器 T0、T1 中断，CPU 响应中断时就用硬件自动清除相应的中断请求标志 TF0 或 TF1。

2）对外部中断 INT0、INT1，若采用边沿触发方式，CPU 响应中断后，内部硬件自动复位中断标志 IE0 或 IE1。

3）对串行中断，CPU 响应中断后并不自动清除相应的中断标志位 TI 或 RI，用户应在串行中断服务程序中用软件清除 TI 或 RI。

4）若 CPU 对中断引脚的信号缺乏控制能力，可以利用单稳态触发器对中断信号进行整形，使其符合要求。

**6. 中断返回**

中断任务处理完成后，CPU 需要返回到中断的地方继续执行。同时需把保存现场内容从堆栈中弹出，恢复寄存器和存储单元的原有内容。正常中断响应时间至少为 3 ~ 8 个机器周期，如果有同级或高级中断服务，将延长中断响应时间。

以上为中断内容的详细描述，关于中断的实际应用在中断章节第 8 章单片机外部中断控制项目中重点介绍。

## 3.4　单片机定时/计数系统

单片机定时/计数器可以实现定时控制、频率测量、脉宽测量、信号发生等功能，其工作方式灵活、编程简单，减轻 CPU 工作负担同时简化单片机外围电路设计。定时/计数器有定时和计数两大功能，对外部事件脉冲计数是计数器；对片内机器周期脉冲计数是定时器。51 单片机有两个定时/计数器 T0 和 T1，在使用过程中与特殊功能寄存器 TMOD 和 TCON 息息相关。

### 3.4.1　定时/计数器结构及工作原理

51 单片机定时/计数器内部结构框图如图 3-13 所示，内部设置两个定时/计数器 T0 和 T1，具有计数器和定时器两种工作方式以及 4 种工作模式，在特殊功能寄存器 TMOD 中都有一个控制位，它选择 T0 及 T1 为定时器还是计数器。

用作定时器时，每个机器周期寄存器加 1，每输入一个脉冲计数值加 1，脉冲数乘以脉冲间隔时间为定时时间；用作计数器时，单片机输入端 P3.4 或者 P3.5 有一个输入脉冲负跳变时加 1，此时脉冲源为间隔不等的外部脉冲发生器，对外部事件计数。

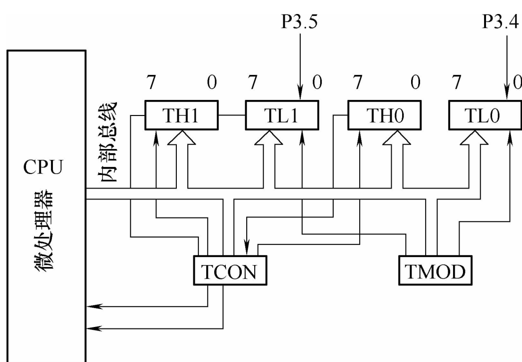



图 3-13　定时/计数器结构框图

图 3-13 中可以看到，定时器 T0 由 TL0（低 8 位）和 TH0（高 8 位）构成，定时器 T1 由 TL1（低 8 位）和 TH1（高 8 位）构成。TCON 则用于控制定时器 T0 和 T1 的启动和停止计数，同时管理定时器 T0 和 T1 的溢出标志等。程序开始时需对 TL0、TH0、TL1 和 TH1 进行初始化，定义其工作方式。

### 3.4.2　定时/计数器特殊控制寄存器 TMOD、TCON

特殊功能寄存器 TMOD 和 TCON 用于控制和确定定时/计数功能和工作模式，寄存器的内容由软件设定，系统复位时所有位均被清零。

**1. 定时/计数器工作方式寄存器 TMOD**

TMOD 低 4 位控制定时器 T0，高 4 位控制定时器 T1，字节地址 89H，只能字节寻址，

TMOD 各位定义见表 3-9：

**表 3-9　TMOD 各位定义**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| GATA | C/$\overline{T}$ | M1 | M0 | GATA | C/$\overline{T}$ | M1 | M0 |
| ←T1 方式字段→ | | | | ←T0 方式字段→ | | | |

M1、M0：工作方式选择位，定时/计数器有 4 种工作方式由 M1、M0 定义见表 3-10：

**表 3-10　M1 和 M0 工作方式**

| M1 M0 | 工作方式 | 功　能 |
|---|---|---|
| 0 0 | 方式 0 | 13 位定时/计数器 |
| 0 1 | 方式 1 | 16 位定时/计数器 |
| 1 0 | 方式 2 | 8 位自动重装载的定时/计数器 |
| 1 1 | 方式 3 | 对于定时器 0，分成两个 8 位计数器，对于定时器 1，停止计数 |

C/$\overline{T}$：定时/计数方式选择位。

C/$\overline{T}$ =1 时为计数方式，内部计数器的输入来自单片机引脚 P3.4 和 P3.5 端口的外部脉冲，负跳变脉冲有效。

C/$\overline{T}$ =0 时为定时工作方式，内部计数器的机器周期脉冲计数，用作定时器。

GATE：选通控制。

GATE =0 时 TR0 或者 TR1 为 1 时，定时/计数器选通工作，与 INT0 或者 INT1 无关。

GATE =1 时 INT0 或者 INT1 引脚为高电平且 TR0 或者 TR1 为 1 时，定时/计数器选通开始工作。

**2. 定时/计数器控制寄存器 TCON**

TCON 是逐位定义 8 位寄存器，可字节寻址也可以位寻址，字节地址为 88H，TCON 各位定义见表 3-11：

**表 3-11　TCON 各位定义**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

1）TF1：定时/计数器 T1 溢出标志。当定时/计数器 1 溢出时，由硬件置位，CPU 响应中断后由硬件自动对 TF1 清“0”。

2）TR1：定时/计数器 T1 运行控制位。靠软件置位或清除，当 TR1 =1 时启动 T1 运行，TR1 =0 则 T1 停止运行。

3）TF0：定时/计数器 0 溢出标志。当定时/计数器 0 溢出时，硬件置位，申请中断，进入中断服务后被硬件自动清除。

4）TR0：定时/计数器 0 运行控制位。软件置位时，T0 工作，清“0”时停止工作。

5）IE1：外部沿触发中断 1 请求标志，检测到在 INT 引脚上出现外部中断信号的下降沿时，硬件置位，请求中断，进入中断程序后被硬件自动清除。

6）IT1：外部中断 1 控制位，IT1 =1 时，后沿触发，IT1 =0 时，低电平触发。

7）IE0：外部沿触发中断 0 请求标志，检测到在 INT 引脚上出现外部中断信号的下降沿

时，硬件置位，请求中断，进入中断程序后被硬件自动清除。

8）IT0：外部中断1控制位，IT0 =1时，后沿触发，IT0 =0时，低电平触发。

### 3.4.3 定时/计数器工作方式

定时/计数器具有定时和计数两种功能，每种功能具有4种工作方式，TMOD控制寄存器通过对M0、M1的设置选择4种不同工作方式，把计数初始值写入TH和TL中控制计数长度；通过对TCON的有关位置数和清零启动定时器工作和停止计数。定时/计数器T1与定时/计数器T0的描述完全一致，下面以T0为例讨论4种工作方式。

**1. 工作方式0**

当M1 M0 =0 0时T0工作在工作方式0模式下，其工作方式0电路结构如图3-14所示，工作方式0是13位定时/计数器，其中TL0的高3位没有使用，计数器由TH0的高8位和TL0的低5位构成。当TL0低5位溢出时向TH0进位，TH0溢出向中断标志TF0进位，申请中断。关于T0计数溢出通过查询TF0是否置位或者产生定时器T0中断获得。

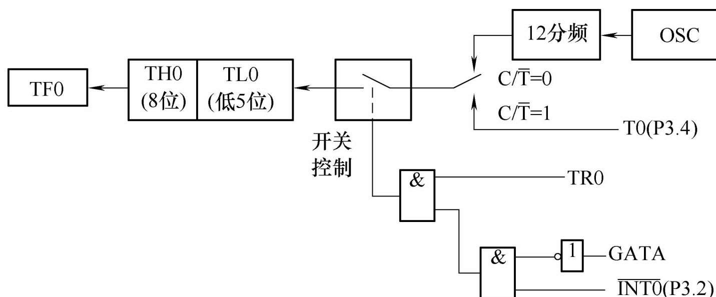



图3-14 T0工作方式0电路结构图

当C/$\overline{T}$ =0时，多路开关接通振荡脉冲的12分频输出，13位计数器依次进行计数。这就是定时工作方式。

当C/$\overline{T}$ =1时，多路开关与引脚（P3.4）相连，外部计数脉冲由引脚T0输入。当外部信号电平发生1到0跳变时，计数器加1，此时T0为外部事件计数器。

当GATE =0时，封锁或门输出常为1，使得引脚P3.2输入信号无效。TR0控制定时器T0开启和关闭，当TR0 =1时，启动定时器T0，TR0 =0则关闭控制开关停止计数。

当GATE =1且TR0 =1时，或门与门全部打开，外部信号电平通过INT0开启或者关闭定时器计数，输入1电平时，计数工作，输入0电平时，停止计数。

（1）作为计数工作

计数范围：$1 \sim 2^{13}$

计数值计算公式：计数值 $=2^{13}-$计数初值

针对T0定时器，其计数初值为TH0高8位和TL0低5位的初始值。

（2）作为定时器工作

定时范围：$1 \sim 2^{13}$机器周期

定时计算公式：定时时间 =（$2^{13}-$定时初值）×机器周期

针对T0定时器，其定时初值为TH0高8位和TL0低5位的初始值。

机器周期时间 = $12/f_{osc}$，其中 $f_{osc}$ 为晶体振荡频率。

**2. 工作方式 1**

当 M1 M0 = 01 时，定时/计数器处于工作方式 1，方式 1 和方式 0 工作模式几乎完全相同，区别就在于工作方式 1 定时器寄存器 TH0 和 TL0 是 16 位计数结构，定时/计数器电路结构如图 3-15 所示。

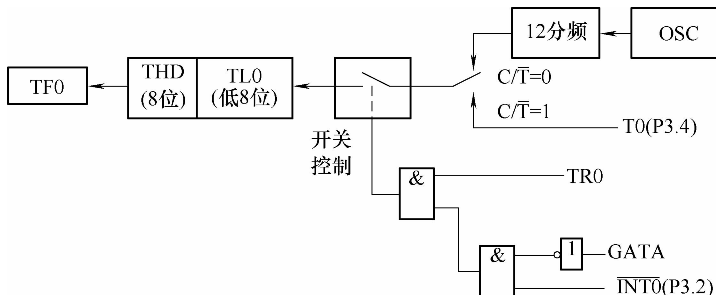



图 3-15　定时/计数器工作方式 1 电路结构图

方式 16 位定时/计数器，TH0 作为高 8 位，TL0 为低 8 位，有关控制状态字（GATA、$C/\overline{T}$、TF0、TR0）和工作方式 0 相同。

（1）作为计数器工作

计数范围：$1 \sim 2^{16}$

计数值计算公式：计数值 = $2^{16}$ − 计数初值

针对 T0 定时器，其计数初值为 TH0 高 8 位和 TL0 低 8 位的初始值。

（2）作为定时器工作

定时范围：$1 \sim 2^{16}$ 机器周期

定时时间计算公式：定时时间 =（$2^{16}$ − 定时初值）× 机器周期

针对 T0 定时器，其定时初值为 TH0 高 8 位和 TL0 低 8 位的初始值。

机器周期时间 = $12/f_{osc}$，其中 $f_{osc}$ 为晶体振荡频率。

**3. 工作方式 2**

当 M1 M0 = 10 时，定时/计数器处于工作方式 2。此时定时器寄存器 TL0 配置为可以自动重装载的 8 位计数器，TH0 作为预置寄存器。TL0 计数溢出时，TF0 置 1 同时 TH0 中的内容重装载到 TL0 中，TH0 的内容由软件预置，重装载后内容不变。电路结构如图 3-16 所示。

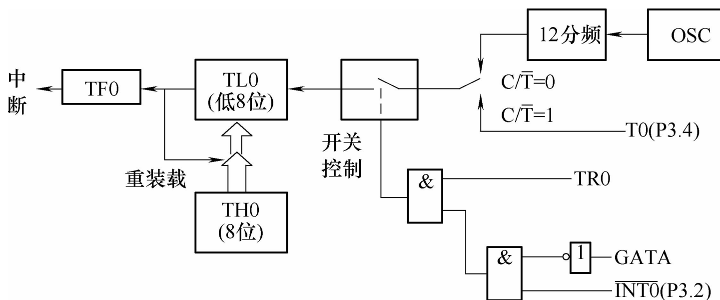



图 3-16　定时/计数器工作方式 2 电路结构图

初始化时，TL0 和 TH0 赋初值，当 TL0 计数溢出时，置位 TF0 的同时把预置寄存器 TH0 中的初值自动加载 TL0，TL0 重新计数。如此反复，这一过程省去了用户程序中重装指令，有利于提高定时精度，但这种工作模式计数值有限，最大只能到 256。所以这种工作方式很适合于连续定时和计数应用。

（1）作为计数器工作

计数范围：$1 \sim 2^8$

计数值计算公式：计数值 $= 2^8 -$ 计数初值

针对 T0 定时器，其计数初值为 TH0 高 8 位和 TL0 低 8 位的初始值。

（2）作为定时器工作

定时范围：$1 \sim 2^{16}$机器周期

定时时间计算公式：定时时间 =（$2^8 -$ 定时初值）× 机器周期

针对 T0 定时器，其定时初值为 TH0 高 8 位和 TL0 低 8 位的初始值。

机器周期时间 $= 12/f_{osc}$，其中 $f_{osc}$ 为晶体振荡频率。

**4. 工作方式 3**

当 M1 M0 = 11 时，定时/计数器处于工作方式 3，此时 T0 分成两个独立的 8 位计数器 TL0 和 TH0。其中 TL0 既可作计数器，也可作为定时器使用，前 3 种工作方式下，定时/计数器的使用完全相同，但是在工作方式 3 下，两个定时/计数器工作是不同的，电路结构如图 3-17 所示。

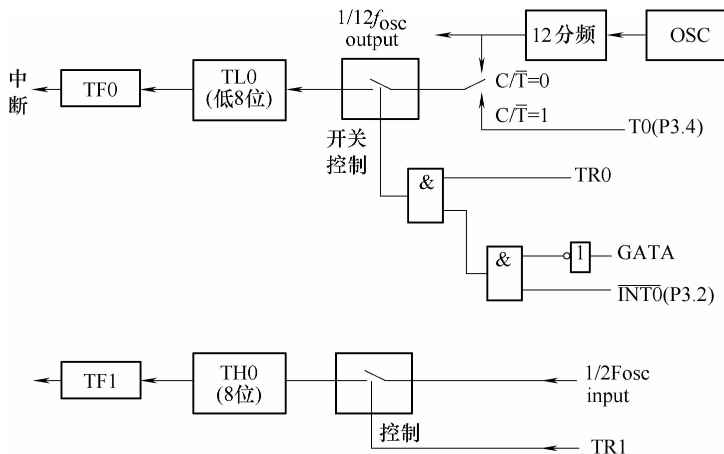



图 3-17　定时/计数器工作方式 3 电路结构图

TH0 规定只用作定时器，由 T1 控制位 TR1 和 TF1 控制，计数溢出置位 TF1、TR1 控制 TH0 定时的启动和停止。由于 TL0 既能作定时器也能作计数器使用，而 TH0 只能作定时器不能作计数器，因此在工作方式 3 模式下，定时/计数器 0 可以构成两个定时器或者一个定时器和一个计数器。

总之，若 T0 设置为工作方式 3，T1 只能设置为方式 0、1、2，因为 T0 占据 TR1 和 TF1，此时 TF1 作为串行口波特率发生器，计数溢出输入串行口，决定串行通信的速率。

经过上述对定时/计数器的讨论，对定时/计数器初始化，首先设定定时器工作方式，设

置 TMOD 工作方式控制字，根据需要给 $C/\overline{T}$ 赋初值确定定时时间或计数初值；设置中断允许寄存器 IE 控制字，开放中断和设定中断优先级；设置 TCON 启动或禁止 $C/\overline{T}$ 运行。关于定时/计数器的实际应用在第 10 章单片机定时控制项目中重点介绍。

## 3.5　单片机串行通信系统

51 单片机具有一个全双工的串行通信端口，能同时进行发送和接收。它可以作为 UART（通用异步接收和发送器）使用，也可作为同步移位寄存器使用。

串行通信都具有多种操作模式，常用于数据通信的传输方式有单工、半双工、全双工和多工方式。

单工方式：通信双方数据按一个固定方向传送，这种传输方式单向传输用途有限，常用于串行口打印数据传输与简单系统间数据采集。

半双工方式：通信双方具有收发器，数据可实现双向传送，但不能同时进行，通信时以某种协议实现收/发开关转换。

全双工方式：通信双方具有收发器，允许双方同时进行数据双向传送，但传输方式线路和设备较复杂。

多工方式：通过使用多路复用器或多路集线器，采用频分、时分或码分复用技术，利用线路资源，实现同一线路上资源共享功能。

### 3.5.1　串行通信结构与原理

51 单片机是一个可编程全双工串行通信口，用作异步通信方式（UART），与串行传送外部设备相连接，或用于通过标准异步通信协议进行全双工 51 单片机多机系统，也可以通过同步方式，使用 TTL 或 CMOS 移位寄存器来扩充 I/O 口。

图 3-18 为 51 单片机串行口结构原理图，通过单片机引脚 RXD（P3.0 串行数据接收端）和引脚 TXD（P3.1 串行数据发送端）与外围电路进行通信，既可以用于网络通信，亦可实现串行异步通信，还可以构成同步移位寄存器使用。51 单片机内部的 SBUF 串行口缓冲寄存器（SBUF）是一个可寻址的专用寄存器，包括接收器和发送器，可实现全双工通信。具有相同的名字和地址空间，因为只有一个能被 CPU 读出数据，另一个只能被 CPU 写入数据，所以不会出现冲突。通信过程只要向发送缓冲器写入数据即可发送数据，从接收缓冲器读出数据即可接收数据。如果在串行口的输入、输出引脚加上电平转换器，将串口转换为标准的 RS232 接口实现与 PC 等设备通信。

### 3.5.2　串行控制与状态寄存器

**1. 串行中断控制寄存器（SCON）**

串行口控制与状态寄存器（SCON）是一个逐位定义的 8 位寄存器，用于定义串行口工作方式以及实施接收和发送控制，字节地址为 98H，位地址 98H ~ 9FH。SCON 各位结构定义见表 3-12。

1）SM0、SM1 串行口工作方式选择位，两个选择位对应 4 种工作方式，其实 $f_{osc}$ 为晶振频率，表 3-13 所示 SM0 和 SM1 工作方式选择位。

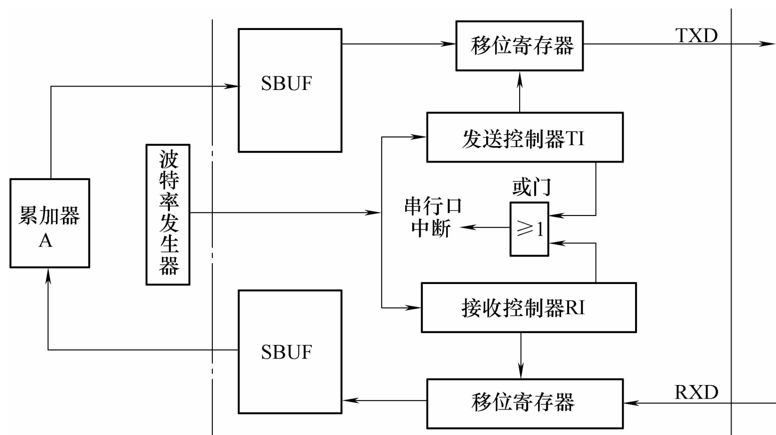



图3-18 串行口结构图

**表3-12 SCON各位结构定义**

| SCON | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
| 位地址 | 9FH | 9EH | 8DH | 9CH | 9BH | 9AH | 99H | 98H |

**表3-13 SM0和SM1串行口工作方式选择位**

| SM0、SM1 | 工作方式 | 功能描述 | 波特率 |
|---|---|---|---|
| 0 0 | 方式0 | 8位同步移位寄存器 | $f_{osc}/12$ |
| 0 1 | 方式1 | 10位UART | 可变 |
| 1 0 | 方式2 | 11位UART | $f_{osc}/64$或$f_{osc}/32$ |
| 1 1 | 方式3 | 11位UART | 可变 |

2）SM2：多机通信控制位。工作方式0中，SM2置0，工作方式1中，若SM2为1，则只有接收到有效停止位时，RI置1被激活。SM2位主要用于工作方式2和方式3，当SM2=1时，只有当接收到第9位数据（RB8）为1时，才把接收到的前8位数据送入SBUF，且置位RI置1发出中断申请，否则会将接收到的数据放弃。当SM2=0时，则接收数据RI不会被激活。

3）REN：允许串行接收位。REN用于控制数据接收的允许和禁止，由软件置位或者清除，当REN=1时，允许串行接收，REN=0时，禁止串行接收。

4）TB8：在方式2或方式3中，为要发送的第9位数据。也可作为奇偶校验位，根据需要由软件置1和清零，在多机通信中，该位用于表示地址帧或者数据帧。

5）RB8：接收到数据第9位。在方式0中不使用RB8。在方式1中，若（SM2）=0，RB8为接收到的停止位。在方式2或方式3中，RB8为接收到的第9位数据。

6）TI：发送中断标志。在方式0中，第8位发送结束时，由硬件置位。在其他方式的发送停止位前，由硬件置位。TI置位既表示一帧信息发送结束，同时也是申请中断，可根据需要，用软件查询的方法获得数据已发送完毕的信息，或用中断的方式来发送下一个数据。

在任何工作方式中，都必须由软件清除 TI。

7）RI：接收中断标志位。在方式 0，当接收完第 8 位数据后由硬件置位。在其他工作方式中，在接收到停止位的中间时刻由硬件置位。RI = 1 时，申请中断，要求 CPU 读数。在工作方式 1 中，SM2 = 1 且未接收到有效的停止位时，不会对 RI 置位。任何工作方式中，都必须由软件清除 RI。

**2. 电源控制寄存器**（PCON）

PCON 主要是为 CHMOS 型 51 单片机实现电源控制而设置的专用寄存器，其中最高位是 SMOD，它是与串行口的波特率设置有关的选择倍增位。单元地址是 87H，表 3-14 所示为 PCON 电源管理寄存器各位定义如下：

**表 3-14　PCON 电源管理寄存器各位定义**

| PCON | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| 位符号 | SMOD | — | — | — | GF1 | GF0 | PD | IDL |

SMOD：串行口波特率倍增位，当 SMOD = 1 时，方式 1 或者 3，波特率 = 定时器 1 溢出率/16；方式 2，波特率 = 定时器 1 溢出率/32；当 SMOD = 0 时，方式 1 或者 3，波特率 = 定时器 1 溢出率/32；方式 2，波特率 = 定时器 1 溢出率/64。

GF1、GF0：通用标志位，用户使用软件置位、复位。

PD：掉电方式位。

IDL：待机方式位，当 IDL = 1，则进入待机工作位。如果 PD = 1、IDL = 1，则进入掉电工作方式。复位时 PCON 所有位均为 0。

## 3.5.3　串行通信工作方式

8051 单片机全双工串行口具有 4 种工作方式，可通过软件编程选择。

（1）方式 0 为移位寄存器输入/输出方式。可外接移位寄存器扩展 I/O 口，也可以外接同步输入/输出设备。其数据传输波特率固定为 $f_{osc}$/12，8 位串行数据由 RXD（P3.0 引脚）输入或输出，同步移位时钟由 TXD（P3.1 引脚）输出。

串行数据发送：CPU 将数据写入发送寄存器时，立即启动发送，内部定时保证写入 SBUF 与激活发送之间有一个完整的机器周期，将 8 位数据以 $f_{os}$/12 的固定波特率从 RXD 输出，低位在前，高位在后。发送完一帧数据后，发送中断标志 TI 由硬件置位。

串行数据接收：当 REN = 1 且接收中断标志 RI 位清除时，启动接收过程。当（RI）= 0 和（REN）= 1 同时满足时，开始接收。当接收到第 8 位数据时，将数据移入接收寄存器，并由硬件置位 RI。

（2）方式 1 为波特率可变的 10 位异步通信接口方式。其传输波特率由定时/计数器 1 的计数溢出率来决定，波特率 = $2^{SMOD}$ ×（定时器 1 溢出率）/32，发送或接收一帧信息为 10 位，包括 1 个起始位 0，8 个数据位和 1 个停止位 1。

串行数据发送：当执行任何一条写 SBUF 的指令时，启动串行数据发送。在串行口由硬件自动加入起始位和停止位，构成一个完整的帧格式，在移位脉冲作用下串行数据从 TXD 引脚输出。发送完一帧数据后 TXD 一直维持在“1”状态下，由硬件置位 TI 通知 CPU 可以发送下一个字符。

数据接收：当 REN =1 且 RI 位清除后，串行口采样 RXD 引脚，若在 RXD 引脚上监测到一个 1 ~0 的跳变，立即启动一次接收过程。只有当 RI =0 且停止位为 1 或者 SM2 =0 时，停止位进入 RB8，8 位数据才能进入接收寄存器，并由硬件置位中断标志 RI；否则信息丢失。所以在方式 1 接收时，应先用软件清零 RI 和 SM2 标志。

（3）方式 2 为固定波特率的 11 位 UART 方式。波特率由 PCON 中的选择位 SMOD 来决定，当 SMOD =1 时，传输波特率为 $f_{osc}/32$，当 SMOD =0 时，传输波特率为 $f_{osc}/64$。发送和接收的一帧信息为 11 位，即 1 个起始位，9 个数据位和 1 个停止位。发送时可编程位 TB8 可赋值 0 或者 1，接收时可编程位进入 SCON 中的 RB8。

数据发送：发送的串行数据由 TXD 端输出一帧信息为 11 位，附加的第 9 位来自 SCON 寄存器的 TB8 位，用软件置位或复位。准备好第 9 个数据位之后，当 CPU 执行一条数据写入 SUBF 的指令时，就启动发送器发送。发送一帧信息后，置位中断标志 TI，其过程与方式 1 相同。

数据接收：方式 2 的接收过程也与方式 1 基本相同，不同之处在于第 9 个数据位，串行口把收到的 8 位数据送入 SBUF，把收到的第 9 位数据送入 RB8。

（4）方式 3 为波特率可变的 11 位 UART 方式。除波特率外，其余与方式 2 相同。

### 3.5.4 波特率设置

波特率是串行通信中的一个重要概念。波特率（bitpersecond）是单位时间里传输的数据位数，单位为波特（或 bit/s）。如数据传送速率是 200 字符/s，每个字符包含 10 位，则传送波特率为 2000 波特。波特率的倒数就是传送每位数据所需要的时间。相互通信的双方必须具有相同的波特率，否则无法成功地完成数据通信。

在串行通讯中，收发双方的数据传送率（波特率）要有一定的约定。51 单片机串行口的 4 种工作方式，方式 0 和 2 的波特率是固定的，而方式 1 和 3 的波特率是可变的，由定时器 T1 的溢出率控制。

**1. 工作方式 0**

波特率由振荡器频率 $f_{osc}$ 确定：波特率 $=f_{osc}/12$

定时器 T1 工作方式 0：溢出所需周期数 =8192 − x，x 为初始值。

**2. 工作方式 1**

波特率由定时/计数器 T1 的溢出率和 SMOD 共同决定，其中：

波特率 $=2^{SMOD}\times$（定时器 1 溢出率）/32

T1 溢出率 = T1 计数率/产生溢出所需的周期数

定时器 T1 工作于方式 1：溢出所需周期数 =65536 − x，x 为初始值。

**3. 工作方式 2**

波特率由振荡器频率 $f_{osc}$ 和 SMOD（专用寄存器 PCON 的最高位）确定。

波特率 $=2^{SMOD}\times f_{osc}/64$，若 SMOD =1 时，波特率 $=f_{osc}/32$；若 SMOD =0 时，波特率 $=f_{osc}/64$。

定时器 T1 工作于方式 2：溢出所需周期数 =256 − x，x 为初始值。

**4. 工作方式 3**

波特率由定时/计数器 T1 的溢出率和 SMOD 共同决定，其中：

波特率 = $2^{SMOD}$ ×（定时器 1 溢出率）/32

T1 溢出率 = T1 计数率/产生溢出所需的周期数

上述讨论中 T1 计数率取决于它工作在定时器状态还是计数器状态。当工作于定时器状态时，T1 计数率为 $f_{osc}$/12；当工作于计数器状态时，T1 计数率为外部输入频率，此频率应小于 $f_{osc}$/24。产生溢出所需周期与定时器 T1 的工作方式、T1 的预置值有关。因为方式 2 为自动重装入初值的 8 位定时/计数器模式，所以用它来做波特率发生器最恰当。

注意：进行单片机电子系统设计时，时钟频率一般选用 11.05926MHz 时，可以获得标准的波特率，所以 51 单片机选用这个看起来很“怪”的晶振。因为方式 0 和 2 波特率是固定传输，方式 1 和 3 波特率是可变的，表 3-15 给出方式 1 和方式 3 常用波特率设置。

**表 3-15　常用波特率设置表**

| 波特率/（kbit/s） | $f_{osc}$ = 12MHz | | | $f_{osc}$ = 11.05926MHz | | |
|---|---|---|---|---|---|---|
| | SMOD | TMOD | TH1 | SMOD | TMOD | TH1 |
| 62.5 | 1 | 20 | FFH | — | — | — |
| 19.2 | — | — | — | 1 | 20 | FDH |
| 9.6 | — | — | — | 0 | 20 | FDH |
| 4.8 | 1 | 20 | F3H | 0 | 20 | FAH |
| 2.4 | 0 | 20 | F3H | 0 | 20 | F4H |
| 1.2 | 0 | 20 | E6H | 0 | 20 | E8H |
| 0.6 | 0 | 20 | CCH | 0 | 20 | D0H |

单片机的串行通信方法较为多样，传统的串行通信方式是通过单片机自带的串行口进行 RS232 方式的通信。串行通信适合远距离数据传送，处于两地的计算机之间，采用串行通信成本低廉，另外关于 51 单片机串行通信应用在第 12 章单片机串口通信项目中重点介绍。

## 3.6　本章小结

本章主要阐述了 51 单片机硬件系统以及体系结构的基础知识，从 51 单片机的引脚配置、存储器、中断、定时/计数器以及串行通信几个方面进行了重点介绍。通过本章的学习，对于初学单片机的读者对单片机的一些基础知识有了入门了解，为后面单片机软件的学习和综合系统设计打下坚实的基础。

# 第 4 章　51 单片机 C 语言程序设计

51 单片机进行软件开发采用的设计语言一般常用汇编语言和 C 语言，本书主要采用 C 语言对 51 单片机进行软件设计，不再介绍汇编语言，同时为了便于初学者能够更好的学习单片机软件设计部分，本章重点对 C 语言中使用的重点知识进行阐述，读者若想更全面了解 C 语言知识，可以参考 C 语言程序设计专门教程。

## 4.1　C 语言简介

C 语言是一种结构化语言，能够产生高效率的紧凑代码，具有丰富的运算符和数据类型，可以实现各类复杂的数据结构。C 语言不需要了解 51 单片机的指令系统，初步掌握存储器结构即可，编译器对寄存器分配、存储器寻址以及数据类型等进行管理，不仅用于系统软件的开发，同时也用于应用软件的开发。另外，程序具有规范的结构，提供的库函数包含许多标准子程序，具有较强的数据处理能力，程序易于模块化、便于移植。

编写 C 语言程序时特点如下：

①一个 C 语言源程序可以由一个或多个源文件组成。

②C 语言程序的扩展名为“. C”。

③每个源文件可由若干函数单元组成，每个函数都是完成某个特殊任务的子程序段。

④一个源程序不论由多少个文件组成，只有唯一一个 main 函数，即主函数。

⑤程序具有可读性。

⑥编程以及程序调试效率高。

C 语言在进行单片机软件设计时使用的词汇有六类：标识符，关键字，运算符，分隔符，常量，注释符等。

**1. 标识符**

C 语言规定，标识符只能是字母（A ~ Z，a ~ z）、数字（0 ~ 9）、下划线（ ）组成的字符串，并且其第一个字符必须是字母或下划线。程序中使用的变量名、函数名、标号等统称为标识符。除库函数的函数名由系统定义外，其余都由用户自定义。在标识符中，大小写是有严格区分的，定义标识符时，取的名字应尽量有直观的意义，便于阅读理解。

**2. 关键字**

关键字是在 C 语言系统中具有特定意义的字符串，用户定义的标识符名字不能与关键字相同。C 语言的关键字分为三类：

（1）类型说明符　用来定义变量、函数或其他数据结构的类型。例如 int、float 等。

（2）语句定义符　表示一个语句的功能意义。如“switch”“if else” 等。

（3）预处理命令字　表示预处理命令的关键字。如例程中用到的“include”。

**3. 运算符**

运算符是编译程序执行特定算术或逻辑操作的符号。C 语言有三大运算符：算术、关系

与逻辑、位操作。

**4. 分隔符**

C 语言中的分隔符有逗号和空格两种。逗号主要用于数据类型说明和函数参数表中分隔变量；空格用于语句各单词之间做间隔符。

**5. 常量**

C 语言中使用的常量可分为数字常量、字符常量、字符串常量、符号常量、转义字符常量等几种。

**6. 注释符**

C 语言的注释符以“/*”开头到“*/”结尾，之间内容即为注释。编译程序时，注释内容不参与编译，只是起到向用户提示或解释程序意义的作用，同时也为编写、调试、维护程序工作提供了便利。

## 4.2　数据结构

具有一定格式的数字或者数值为数据，数据的不同格式为数据类型，按照一定的数据类型进行排列、组合、架构称为数据结构。

### 4.2.1　数据类型

在 C 语言中，数据类型分为基本类型，构造类型，指针类型，空类型四大类，其中基本类型主要包括整型（int）、字符型（char）、实型也称为浮点型（float）和枚举类型（enum）；构造类型主要包括数组类型、结构体类型和共用体类型；指针类型是 C 语言编写程序的灵魂，也是初学者重点学习 C 语言的部分；空类型主要针对 C 语言编写函数时，调用函数后不需要向调用者返回函数值所采用的编程方式。本节对经常用到的数据类型做简单的介绍。

**1. 整型**（int）

int 类型长度为两个字节，分为有符号整型（signed int）和无符号整型（unsigned int），默认值为 signed int 类型。

**2. 长整型**（long）

long 类型长度为 4 个字节，分为有符号长整型（signed long）和无符号长整型（unsigned long），默认值为 signed long 类型。

**3. 字符型**（char）

char 类型长度为 1 个字节，通常用于定义处理字符数据的变量或常量。分为有符号字符类型（signed char）和无符号字符类型（unsigned char），默认值为 signed char 类型。

**4. 浮点型**（float）

float 类型的长度为 32 位，一般数学表达式采用浮点数据类型。

**5. 枚举类型**（enum）

enum 类型是指将变量的值一一列出来，变量的值只限于列举出来的值的范围内。

**6. 构造数据类型**

对一个或多个数据类型用构造的方法定义。一个构造类型的值可以由若干个“成员”

或“元素”组成。“成员”是一个基本数据类型或构造类型。在 C 语言程序设计中，构造类型包括三类：

①数组类型：相同数据类型的元素按一定顺序排列的集合，有限个类型相同的变量用一个名字命名，然后用编号区分他们变量的集合，这个名字成为数组名，编号成为下标。组成数组的各个变量成为数组的分量，也称为数组的元素。

②结构体类型：是一种构造类型的数据，将若干个不同类型的数据变量有序地组合在一起而形成的一种数据集合体。组成该集合体的各个数据变量为结构成员，整个集合体使用一个单独的结构变量名。

③共用体类型：是 C 语言一种构造类型的数据结构，所占内存空间的长度是其中最长的成员长度，各个成员的数据类型以及长度从同一个地址开始存放，不同的变量分时使用一个内存空间，有效提高内存的利用效率。

**7. 指针类型**

指针是一个特殊的变量，它里面存储的数值被解释成为内存里的一个地址，也是 C 语言的精华所在，C 语言区别其他程序设计语言的主要特点是处理指针时所表现出的能力和灵活性，正确使用指针类型数据，可以有效的表示复杂的数据结构，直接处理内存地址，合理使用数组。

**8. 空类型**

函数在被调用完成后，通常会返回一个函数值，调用后并不需要向调用者返回函数值，此时，将这种函数定义为“空类型”，其类型说明符为 void。

## 4.2.2 常量与变量

对于基本数据类型量，根据变量值在程序执行过程中是否发生变化，又可分为常量和变量两种。在程序运行过程中值不能改变的量称为常量，而值可以不断变化的量称为变量。

常量——与变量相对应，在程序执行的过程中，其值不能发生改变。常量的数据类型有位型、整型、浮点型、字符型和字符串型。

位型常量为一位二进制值。

整型常量可以为十进制，如 1，3，30 等；亦可以为十六进制以 OX 开头，如 OX2F、OX5A 等。

浮点型常量有十进制和指数两种形式。十进制由数字和小数点组成，如 34.44，0.0 等，指数形式［±］数字［.数字］*e*［±］数字，［ ］中的内容为可选项，其中内容根据具体情况可有可无，但是其余部分必须有，如 8e4 等。

字符型常量是单引号内的字符，如‘f’，无法显示的控制字符可以在该字符前面加一个反斜杠“\”，组成专用转义字符。转义字符用来表示那些用一般字符不便于表示的语句代码。表 4-1 为 C 语言常用的转义字符描述。

**表 4-1　C 语言常用的转义字符描述**

| 转义字符 | 转义字符的意义 | 转义字符 | 转义字符的意义 |
|---|---|---|---|
| \n | 回车换行 | \b | 退格 |
| \t | 横向跳格（跳到下一输出区） | \r | 回车 |

（续）

| 转义字符 | 转义字符的意义 | 转义字符 | 转义字符的意义 |
|---|---|---|---|
| \ f | 走纸换页 | \ a | 鸣铃 |
| \\ | 反斜线符" \ " | \ ddd | 1 ~3 位八进制数所代表的字符 |
| \ ’ | 单引号符 | \ xhh | 1 ~2 位十六进制数所代表的字符 |
| \ ” | 双引号符 | | |

字符串常量由双引号内的字符组成，如“student”等，若双引号内没有字符时为空字符串。在 C 语言中字符串常量作为字符类型数组来处理，在存储字符串时系统会在字符串尾部加上 \ 0 转义字符，作为字符串的结束符。

注意：字符串常量和字符常量区别为“F”：字符串常量；‘F’：字符常量。字符串常量存储时比字符常量多占用一个字节为存储结束符 \ 0。

符号常量在使用之前必须先定义，其一般形式为

#define 标识符 常量

其中#define 是一条编译预处理命令（预处理命令都以" #" 开头），称为宏定义命令，它的功能是把该标识符定义为其后的常量值。习惯上用大写字母来表示符号常量的标识符，用小写字母表示变量标的识符。使用符号常量的优点：如当程序中很多地方都要用到这个变量，而数值又需要经常做改动，这时使用符号常量方便修改。

变量——由两部分构成，一个是变量名，一个是变量值。每个变量都有一个变量名，在内存中占据一定的存储单元，并在该内存单元中存放该变量的值。程序中使用变量须先用标识符作为变量名，并指出所用的数据类型和存储模式。

**1. 整型变量**

变量定义的一般形式为

类型说明符，变量名标识符，变量名标识符,...；

例如：

int a，b，c；（a，b，c 为整型变量）

unsigned int a，b；（a，b 为无符号整型变量）

定义变量时，允许在一个类型说明符后，同时定义多个相同类型的变量。各变量名之间用逗号间隔，类型说明符与变量名之间至少用一个空格间隔。

**2. 实型变量**

实型变量分为单精度（float 型）和双精度（double 型）。

其中单精度型占 4 个字节内存空间，双精度型占 8 个字节内存空间。

实型变量定义方法与整型变量定义方式相同，

例如：

float a，b；（a，b 为单精度实型量）

double a，b；（a，b 为双精度实型量）

**3. 字符变量**

字符变量用来存储单个字符，定义处理字符数据的变量。

字符变量定义与整型和实型方式相同，

例如：

char a，b；（a，b为字符型变量）

**4. 位变量**

sbit用于定义可位寻址的对象，如访问特殊功能寄存器中的某位。位变量定义如下：

sbit 位变量名 = 位地址

例如：sbit P1－1 =0X91；将位的绝对地址赋值给位变量。

sbit 位变量名 = 特殊功能寄存器名^位位置

例如：sbit P1－1 = P1^1；指定位变量名所在位置。

## 4.3 运算符与表达式

运算符是完成某种特定运算的符号，表达式由运算符以及运算对象所组成的具有特定含义的式子，任意一个表达式后面加一个分号“;”构成表达式语句。按照运算符在表达式中所起的作用分为算术运算符、关系运算符、关系运算符、逻辑运算符和位运算符。

### 4.3.1 运算符分类

C语言的运算符分7类：

1）算术运算符：用于数值比较运算。

2）关系运算符：用于比较运算。

3）逻辑运算符：用于逻辑运算。

4）位操作运算符：参与运算的量按二进制位进行运算。

5）赋值运算符：用于赋值运算。

6）条件运算符：<表达式1>？<表达式2>：<表达式3>含义为若<表达式1>值为真，则条件表达式取<表达式2>的值；否则取<表达式3>的值。

7）指针运算符：用于取内容“*”和取地址“&”两种运算。

### 4.3.2 算术运算符与表达式

（1）基本的算术运算符

①加法运算符“+”：如a+b。

②减法运算符/负数值符号“－”：如a－b或负数－8。

③乘法运算符“*”：如a*b。

④除法运算符“/”：如a/b。

⑤求余运算符（模运算符）“%”：如7%4 =3，即7除以4后，余数为3。

（2）自增1运算符“++”，其功能使变量的值自增1；自减1运算符为“--”，其功能使变量的值自减1，运算具有左右结合特性有4种形式：

①++i：i自增1后再参与其他运算。

②--i：i自减1后再参与其他运算。

③i++：i参与运算后，i的值再自增1。

④i--：i参与运算后，i的值再自减1。

i ++ 和 i -- 在程序设计时容易出错，尤其程序复杂对于初学者容易搞混淆，注意分辨。

### 4.3.3　关系运算符与表达式

关系运算符用于比较运算，关系表达式通常是用来判别某个条件是否满足。包括大于（ > ）、小于（ < ）、大于等于（ ≥ ）、小于等于（ ≤ ）、等于（ == ）和不等于（ ! = ）六种。关系运算符的运算结果只有“0”和“1”两种，当结果为“1”，则表示指定的条件满足，不满足时的结果为“0”。

关系表达式的一般形式为

表达式 关系运算符　表达式

例如：a + b ≥ c

关系表达式允许出现嵌套表达式。

例如：x ! = ( a > b )

### 4.3.4　逻辑运算符和表达式

逻辑运算符用于逻辑运算，C 语言提供三种逻辑运算符：

①&&　“与”运算

② ||　“或”运算

③!　“非”运算

其中逻辑运算符优先级从高到低排列如下：!（逻辑非）→&&（逻辑与）→ ||（逻辑或），其中，逻辑非的优先级最高，逻辑或的优先级最低。

例如：4 > 0 && 9 > 2

因为 4 > 0 为真，9 > 2 也为真，两边同时满足真，所以它们相“与”的结果也为真。即为“1”。

### 4.3.5　赋值运算符和表达式

变量赋初值 -- C 语言可以给变量赋初值，称之为变量的初始化操作。

变量初始化赋值的形式为

类型说明符 变量 1 = 值 1，变量 2 = 值 2，变量 3 = 值 3…；

例如：

```
int a =3;
char channel1 ='A', channel2 ='B';
```

C 语言在变量定义中不允许出现多个“ = ”赋值符号，如 a = b = c = 5 是非法的。

复合赋值运算符—在赋值运算符“ = ”的前面加上其他运算符构成复合赋值运算符，如下：

+=　加法赋值运算符

—=　减法赋值运算符

*=　乘法赋值运算符

/=　除法赋值运算符

%=　取模赋值运算符

>>=　右移位赋值运算符

<<=　左移位赋值运算符

& =　逻辑与赋值运算符

| =　逻辑或赋值运算符

^ =　逻辑异或赋值运算符

⎵=　逻辑非赋值运算符

复合赋值表达式一般形式为

变量　运算符 = 表达式　等效：变量 = 变量 运算符 表达式

例如：a + =7 等效于 a = a +7；a% =b 等效于 a = a%b

对于初学者复合赋值的写法不太习惯，但是该写法有利于编译程序，提高编译效率。

### 4.3.6　位运算符与表达式

位运算符是按位对变量进行运算，不改变参与运算的变量的值，注意：位运算符不能用来对浮点型数据进行操作。位运算符的优先级从高到低顺序为

①按位取反（⎵）

②左移（<<）和右移（>>）

③按位与（&）

④按位异或（^）

⑤按位或（|）

表 4-2 给出位运算操作真值表。

**表 4-2　位运算操作逻辑真值表**

| a | b | ⎵a | ⎵b | a&b | a \| b | a^b |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |

## 4.4　函数使用

C 语言中的函数就是用来实现模块化程序设计的工具，用户可以把自己的程序写成一个一个相对独立的函数，函数中不能定义其他函数，但是函数可以互相调用，函数可以自己调用自己成为递归调用。函数调用的一般规则是主函数可以调用其他普通函数，普通函数之间可以互相调用，但是普通函数不能调用主函数。

### 4.4.1　C 语言程序的基本结构

C 语言程序的一般组成结构如下：

预处理命令　　include < >

全局变量声明

```
main( )
{                         //主函数体
    局部变量申明
    执行语句
}
函数 1(形参表)        //函数 1
形参申明
{     局部变量申明
      执行语句
}
…
函数 n(形参表)      //函数 n
形参申明
{     局部变量申明
      执行语句
}
```

C 程序的执行从 main( ) 函数开始，不管物理位置上这个 main( ) 在什么地方，主函数通过直接书写语句和调用其他功能子函数来实现有关功能，这些功能子函数可以是 C 语言本身提供的库函数，也可以是用户自己编写的函数。对于初学者函数的学习是 C 语言程序设计的重点。

### 4.4.2　函数定义

函数由类型说明符、函数名、参数表和函数体四部分组成。函数名区分大小写；参数表是用括号括起来的若干参数，参数之间用逗号隔开；函数体用大括号括起来的可执行语句，语句与语句之间用分号隔开，最后一条语句为 return(主函数中可省略)，每个函数都返回数值。

**1. 无参函数定义**

```
类型标识符 函数名()
    {  类型说明
      可执行语句
}
```

其中“类型标识符”指明函数返回值类型。函数名由用户自定义，后面是空括号，代表没有函数参数，即代表无参函数，注意空括号不可以省略。

大括号中的内容被称为函数体。注意：在函数体中有类型说明，函数体中声明的各种对象，只能在函数体内有效。

一般情况下，无参函数没有返回值，因此可以将函数的类型标识符写成“void”。

例如：

```
void print ( )
{
```

```
    printf (“this is a desk. \n ”);
}
```

上述语句中 print 为一个无参函数的函数名，并且没有返回值。一旦被调用输出“this is a desk. ”字符串。

**2. 有参函数定义**

类型标识符 函数名（形参表）

```
    {类型说明部分
     执行语句
    }
```

有参函数比无参函数多了两项，①形参表，②形参类型说明。在该列表中列出的形参称为形式参数，简称为形参，它可以是各种类型的数据，各种参数之间用“,”号分隔。函数被调用时，主调函数将通过实际参数，简称实参，传递实际的值给这些形参。

例如：定义一个有参函数，求两个数中的大数，

```
int big( int x, int y)
{
    if ( x > y) return x;
    else return y;
}
```

程序中第一行定义了一个函数，函数名为 big 函数，其返回类型定义为整型（int）型，x，y 为函数的形参，并且定义 int 类型。x，y 的值由主调函数在调用时传送过来。当 big 被调用时，主调函数将通过实参将实际的值传给形参 x 和 y。函数体中的 if 语句用来判断，如果 x > y，使用 return 语句返回 x 的值，否则就返回 y 的值。在这里有几点需要注意：

1）有返回值的函数至少应有一个 return 语句，C 语言中，函数可以放在主函数 main 之前也可以 main 之后。

2）形参只出现在函数定义中，在函数体中可以被使用，但在函数体外不能使用；实参只出现在主调函数中，在调用函数时，把实参的值传递给被调函数的形参，从而实现主调函数向被调函数的数据传递。

3）函数定义时没有写明类型标识符，则默认为整型。

4）函数返回值 return（表达式），其中“表达式”为函数返回主调函数的值，必须和函数的类型标识符一致。

5）函数不需要返回值，函数类型标识符则可以写作“void”，表示该函数没有返回值。

### 4.4.3 函数调用

所谓的函数调用就是在一个函数体中引用另外一个已经定义的函数，前者为主调函数，后者为被调用函数。C 语言中，函数调用一般形式如下：

函数名（实际参数表）

其中“函数名”为被调用的函数

“实际参数表”，即实参表是可以没有的，表示无参函数。实参表可以是任何类型的数据，可以是常量，变量及表达式。各参数之间用“,”分隔。实际参数的作用是将它的值传

递给被调用函数中的形式参数。

C 语言中采用三种方式完成函数调用分别为①函数语句调用②函数表达式调用③函数参数调用。

**1. 函数语句调用**

主调函数中将函数调用作为一条语句，例如：

max( )；　//无参调用，不要求被调函数返回确定数值。

**2. 函数表达式调用**

主调函数中将函数调用作为一个运算对象直接出现在表达式中，例如：

C = min(a,b)；　//赋值语句,把 min 返回值赋给变量 C。

这种调用方法要求被调函数返回一个确定值。

**3. 函数参数调用**

函数作为另一个函数调用的实际参数出现。这种情况是把该函数的返回值作为实参进行传送，因此要求该函数必须是有返回值。例如：

M = max(a,max(b,c))；　//max(b,c)是一次函数调用，返回值作为函数 max 调用的实参。M 的值为变量 a、b、c 中的最大值。该段代码可以看到函数在调用过程中嵌套函数调用，既调用一个函数过程中又调用另一个函数。在函数调用过程中有几点注意事项：

①在函数定义之前，在主函数外部预先说明各个函数的类型，则不需要在后面的主调函数中再对其进行说明。例如：

```
int   ann(int x);
char circle(int y);
main()
{
语句;
}
int ann(int x)
{
语句;
}
char circle(int b)
{
语句;
}
```

通过对 ann 函数和 circle 函数预先说明。因此在以后各主调函数中不用对 ann 和 circle 函数说明,可直接调用。

②对系统库函数的使用或者不在同一文件中自定义函数，必须使用 include 命令，把函数相应的头文件包含到文本中。

### 4.4.4　函数的嵌套调用和递归调用

函数嵌套调用—C 语言不允许嵌套函数定义，但是允许一个函数的定义中出现对另一个

函数的使用，既函数的嵌套调用。通过程序结构了解函数嵌套的编写。如嵌套函数样例程序如下：

```
main( )              /*定义主函数
{
    max( );      /*在主函数中调用 max( )函数*/
    }
max( )
{
    min( );      /*在 max( )函数中调用 min( )函数*/
    printf("hello\n");
  }
min( )
{
    printf("the min is a\n");
    }
```

程序执行结果为

the min is a

hello

程序运行过程先执行主函数 main，main 函数调用 max 函数，在执行 max 函数时，max 函数调用 min 函数，min 函数执行完毕后输出 the min is a，再返回 max 函数的转出点继续执行剩余代码，max 函数执行完毕后输出 hello，最后再返回 main 函数的转出点继续执行剩余代码。

函数递归调用—函数在它的函数体内，调用它自身称为函数的递归调用。C 语言是允许递归调用的。在递归调用中，主调函数又是被调函数，执行递归函数将反复调用自身，每调用一次就进入新的一层，一旦条件满足（通过 if 语句书写条件）终止递归调用，返回结果。

函数递归调用样例程序如下：计算 x 的 n 次方（x 和 n 均为正整数）。

```
main( )
{
    int a,y;
    scanf("%d,%d",&a,&y);
    printf("%d**%d=%d\n",a,y,power(a,y));
  }
power(int x,int n)
{
    int p;
    if(n>0)
    p=power(x,n-1)*x;
    else
```

```
    p = 1;
  return(p);
  }
```

在该例中，power 函数通过 power（x，n－1）直接调用自身，通过变量 n 控制程序调用的次数，从而实现计算 x 的 n 次方。

递归调用的优点是使程序看上去简洁明了，可以使一些原本复杂的程序简化，但是由于函数反复调用占用较大的存储空间，一旦递归调用次数较多，则程序编译效率低，需要优化程序结构。

## 4.5　数组与指针

数组是一组有序数据的集合，数组中每一个数据都属于同一个数据类型，数组类型的所有元素属于同一种类型，并且按顺序存放在一个连续的存储空间。数组中各个元素用数组名和下标唯一确定。C 语言书写程序时需要先定义，然后才能使用。数组可以是一维数组、二维数组或者多维数组。一维数组只有一个下标，多维数组有两个以上下标。

### 4.5.1　数组

**1. 一维数组**

一维数组定义为：

数据类型 数组名 [常量表达式]；

其中：

数据类型：数组中各数据元素的类型。

数组名：整个数组的标识，命名方法和变量命名方法是一样的，数组名是所分配空间首地址的标识。

常量表达式：表示数组的长度，既数组中元素个数，必须用“[ ]”括起，方括号里的数不能含有变量。

例如：

int student[20]；　//说明整型数组 student 中有 10 个学生。

学习数组时有几点需要注意：

① 数组的下标是从 0 开始，如 int student[20]，下标就是从 student[0]到 student[19]。

② 数组名不能与其他变量名同名。

③C 语言允许同一个类型说明中，说明多个数组和多个变量。例如：int a，b，c，s[20]。

④ 数组定义后，数组中各元素共用一个数组名，通过下标区分各个元素。

**2. 二维数组**

二维数组定义形式：

数据类型 数组名[常量表达式 1][常量表达式 2]

其中常量表达式 1 表示第一维大小，常量表达式 2 表示第二维大小。

定义一个 3 行 3 列共 3 ×3 =9 个元素的整型数组，可以采用二维数组定义

```
int a[3][3];
```

其中数组各个元素为

a[0][0], a[0][1], a[0][2]

a[1][0], a[1][1], a[1][2]

a[2][0], a[2][1], a[2][2]

二维数组赋值时，可以分段赋值也可以连续赋值。

例如：对数组 a[3][3]赋值：

① 按分段进行赋值为

```
int a[3][3] = { {8, 7, 9}, {7, 6, 7}, {7, 8, 7}};
```

② 按连续进行赋值为

```
int a[4][3] = { 8, 7, 9, 7, 6, 7, 7, 8, 7};
```

其结果是一样的。

### 4.5.2 指针

指针是 C 语言的知识精华，对于初学 C 语言读者应当深入学习和掌握指针。更多关于指针知识的学习可以参考 C 语言相关技术书籍。

**1. 指针定义**

指针用于存放变量的地址，该地址是另一个变量在内存中存储的位置。指针它本身也是一种变量，和其他变量一样，要占有一定数量的存储空间，用来存放指针值(即地址)。

指针定义一般形式为

数据类型 *指针变量名;

其中，数据类型：表示该指针变量所指向变量的类型。

指针变量名：定义指针变量的名字。

例如：

int *point; 指针变量 point 是指向 int 类型变量的指针。

注意区分变量的指针和指针变量：变量的指针是变量的地址，而一个指针变量存放的内容是另一个变量在内存中的地址，拥有这个地址的变量为该指针变量所指向的变量。每一个变量都有自己的指针(称之地址)，每一个指针变量指向另一个变量。

例如：整型变量 x 的地址 30H 存放在指针变量 point 中，可用 *point 表示指针变量 point 指向的变量，即 *point 表示变量 x。

**2. 指针运算符 & 和 ***

指针变量中只能存放地址，基本的运算符是 & 和 *。

(1) &——取地址运算符，返回变量的内存地址。只能用于一个具体的变量或数组元素，不可用于表达式。

例如：

```
int *p;
int t;
p = &t;
```

说明将整型变量 t 的地址赋值给指针变量 p。

（2） * —— 指针运算符,返回地址中的变量值。

例如:

```
int *p;   //指针变量定义
int t;
int x;
p=&t;   //&t 中的 & 为取 t 的地址赋值给 p。
x=*p;   //将指针变量 p 指向的变量值赋给 x。
```

在指针赋值时注意几点:

①指针使用之前,未初始化的指针变量不可以使用。

②赋值语句中,变量的地址只能赋给指针变量本身。

**3. 指针数组**

（1）指针数组的定义　一个数组里面的元素是指针变量称为指针数组。指针数组的一般形式为

类型说明符 *指针数组名[元素个数];

例如:

```
int *p[10];
```

p[10]是一个指针数组,含有 10 个指针,并指向 int 型数据。

（2）指针数组初始化　只有全局或静态的指针数组才可进行初始化，另外，不能用局部变量的地址去初始化静态指针。

例如：指针数组

```
main()
{
  static int b[2][3]={{1,2,3},{4,5,6}};
  static int *pb[]={b[0],b[1]};/*指针数组指向行首 */
  int i,j;
  for(i=0;i<2;i++){
    for(j=0;j<3;j++)
        printf("b[%d][%d]=%d",i,j,*(pb[i]+j));
    printf("\n")
  }
}
```

执行结果:

```
b[0][0]=1 b[0][1]=2 b[0][2]=3
b[1][0]=4 b[1][1]=5 b[1][2]=6
```

程序将一个二维数组 b[2][3]分解成两个一维数组。它们的首地址,分别为 b[0]和 b[1],也可以理解为第一行的首地址,和第二行的首地址,并被赋给指针 pb[0]和 pb[1]。

## 4.6 程序设计语句

从程序流程的角度看，依据结构化程序设计，结构清晰、层次分明易于修改和阅读。程序结构可以分为顺序结构、分支结构和循环结构三种基本形式。

**1. 顺序结构**

顺序结构是最简单，最基本的编程结构，如图 4-1 所示。程序由低地址向高地址顺序执行指令代码。程序按照书写顺序依次执行。

**2. 选择结构**

选择结构是对给定的条件进行判断，依据判断结果决定执行哪个分支，如图 4-2 所示，若条件判断为真，执行语句 1，否则执行语句 2。

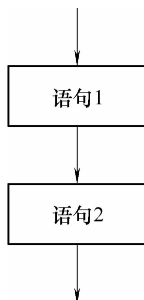



图 4-1 顺序结构

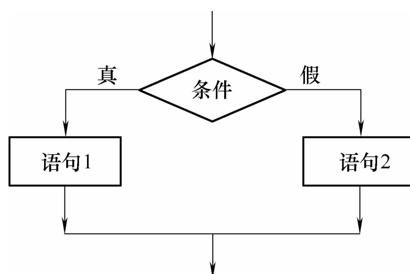



图 4-2 选择结构

**3. 循环结构**

循环结构是在给定条件成立时反复执行某段程序，有两种形式：①当型循环结构；②直到型循环结构。

当型循环结构如图 4-3 所示。条件成立时，反复执行语句，直到条件不满足为止。

直到型循环结构如图 4-4 所示。先执行语句操作，再进行判断，若为假，再执行语句，直到条件为真为止。

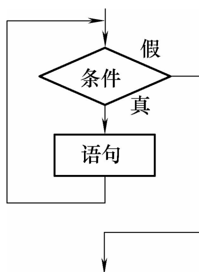



图 4-3 当型循环结构

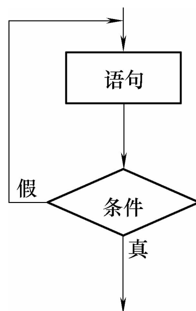



图 4-4 直到型循环结构

C 语言流程控制语句可分成三类：

选择语句：if 语句和 switch 语句。

循环语句：for 语句和 while 语句。

转移语句：break 语句和 continue 语句。

### 4.6.1　选择语句

**1. if 语句**

条件语句又被称为分支语句，由关键字 if 构成。根据给定的条件进行判断，决定执行某个分支程序段。C 语言提供了 3 种形式的条件语句：

①基本形式：

```
if（表达式）
    语句；
```

如果表达式是真就执行后面的语句，否则就不执行。

②if-else 形式：

```
if（表达式）
  语句 1；
else
  语句 2；
```

如果表达式的值为真，则执行语句 1，否则执行语句 2。

③if-else-if 形式：

当有多个分支选择时，采用 if-else-if 语句，形式为

```
    if（表达式 1）
    语句 1；
else  if（表达式 2）
    语句 2；
else  if（表达式 3）
    语句 3；
    …
else  if（表达式 m）
    语句 m；
else
    语句 n；
```

先判断表达式 1 的值，如果为真，则执行语句 1，如果表达式 1 的值为假，则再判断表达式 2 的值，如果表达式 2 的值为真，则执行语句 2，否则继续判断表达式 3 的值，依次判断表达式的值，当出现某个值为真时，则执行后面对应的语句，语句执行完后跳到整个 if 语句之外继续执行程序代码。如果所有的表达式都为假，那么执行语句 n，即最后一个 else 后面的语句，然后再继续执行后面的程序代码。

if 语句若想在满足条件分支时执行多条语句，必须把语句用“{}”括起来，每一条语句末尾需要加上“;”，分号应加在“}”之内，而不能加在“}”外面。

例如：

```
    if(x>0)
```

```
        {
            i ++ ;
            printf( " x > 0" ) ;
        }
    else
        {
            i -- ;
            printf( " x < =0" ) ;
        }
```

程序中如果 x >0，变量 i 自加 1，输出“x >0”；如果 x <0 或 x =0，变量 i 自减 1，输出“x <=0”。

如果 if 语句中的执行语句包含 if 语句，则构成 if 语句嵌套，采用嵌套结构实质上是为了进行分支选择。在嵌套内的 if 语句又是 if-else 型，嵌套内的语句可以是多个 if 和多个 else 重叠的情况，对于初学者注意 if 和 else 配对问题。

**2. switch 语句**

一般程序有过多的分支时，由于分支太多使程序看起来比较混乱，则使用 switch 语句，可以使程序结构清晰。switch 语句一般形式为

```
   switch( 表达式)
   {
        case 常量表达式 1:  语句 1;break;
        case 常量表达式 2:  语句 2; break;
        …
        case 常量表达式 n :  语句 n; break;
        default          :  语句 n +1;
   }
```

意义是计算 switch 后面表达式的值，并将其作为条件与 case 后面的各个常量表达式的值相比，如果相等则执行 case 后面的语句，再执行 break（间断语句）语句，跳出 switch 语句结构；如果 case 后面没有和条件相等的值时就执行 default 后的语句。如果没有符合的条件，不做任何处理，可以不写 default 语句，default 语句只是程序不满足所有 case 语句条件情况下的一个默认情况执行语句。

使用 switch 语句时注意以下几点：

①case 后的各常量表达式的值是不一样的，否则会出现错误。

②在 case 后，允许出现多条语句，可以不用 {} 括起来。

③各 case 和 default 语句位置的先后顺序可以改变，而不会影响程序执行结果。

④default 子句可以省略不写。

程序举例：用户输入运算数和四则运算符，输出计算结果。

```
main( )
{
    float a,b;
```

```
    char m;
    printf("input expression: a+(-,*,/)b \n");
    scanf("%f%c%f",&a,&m,&b);
    switch(m)
    {
        case '+': printf("%f\n",a+b);break;
        case '-': printf("%f\n",a-b);break;
        case '*': printf("%f\n",a*b);break;
        case '/': printf("%f\n",a/b);break;
        default: printf("input error\n");
    }
}
```

本程序通过输入“+”、“-”、“*”、“/”四个字符，输出对应的运算数据结果为 a+b、a-b、a*b 和 a/b，若输入的字符不是上述四个字符则输出 input error，可见多分支程序用 switch 语句可轻松实现。

### 4.6.2　循环语句

循环结构在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件为循环条件，反复执行的程序段为循环体。用循环语句书写重复执行语句，不但使程序结构简洁，查看代码显示直观，而且使其编译的效率大大提高。

**1. for 语句**

C 语言中的 for 语句使用非常灵活，不仅可以用于循环次数已定的情况，而且可用于循环次数不确定，只给出循环结束条件的情况。一般形式为

```
for（初值设定表达式 1；循环条件表达式 2；条件更新表达式 3）
{
  语句；    /*循环体*/
}
```

for 语句执行过程如下：

①先执行初值设定表达式 1 的值作为循环控制变量的初值。

②然后求循环条件表达式 2 的值，当满足循环条件时执行循环体语句并计算更新表达式 3。

③然后根据更新表达式 3 的计算结果判断循环条件 2 是否满足。

④一直进行到循环条件表达式 2 的结果为假时，退出循环体。

例如：用 for 语句计算 s=1+2+3+4+……+99+100 的结果：

```
Void main( )
{
    int n,s=0;
    for(n=1; n<=100; n++)
     s=s+n;
```

```
    printf("s=%d\n",s);
}
```

for 语句先给变量 n 赋初值 1，判断 n 是否小于或等于 100，如果为真，则执行语句“s = s + n”，然后 n 自增 1。然后再重新判断，直到条件为假，即 i > 100 时，循环结束。

对于使用 for 语句，要注意以下几点：

①for 语句中各表达式可以省略，但是表达式之间的分号间隔符号不能省略。

②省略“循环条件表达式 2”，如程序中不做相应的处理，此时应在循环体内结束循环，否则将变成为死循环。

③循环体可以是空语句。

for 语句可以构成多重循环，既循环嵌套。所谓循环嵌套，指循环体里面包含了另一个完整的循环。

例如：

```
for(i=1;i<=1000;i++)
  {s=0;
  for(j=1;j<i;j++)
      if(i%j==0)
        s+=j;
  }
```

程序中使用两层 for 循环嵌套。其中外层循环变量为 i，控制数据的取值范围；内层循环变量为 j，内层循环的循环体只有一条语句用于求对应每一个 i 所有的因子和。

**2. while 语句**

while 语句一般形式为

```
while(表达式)
{
  语句;/*循环体*/
}
```

其中表达式为循环条件，语句为循环体。判断表达式是否为真，若为真则执行后面的语句，执行一次完成之后再次回到 while 后面的表达式，进行判断，如果为真，则重复执行语句，否则跳出循环。当条件一开始就为假时，那么 while 后面的循环体一次都不会被执行就退出整个循环。

例如：用 while 语句求 s = 1 + 2 + 3 + 4 + …… + 99 + 100。

```
main()
{
    int i,sum=0;
    i=1;
    while(i<=100)
      {
         sum=sum+i;
         i++;
```

```
    }
    printf("%d\n",sum);
}
```

程序在执行过程中注意几点事项：

①如果第一次进入循环时，while 后圆括号内表达式的值为 0，循环一次也不执行。在本程序中，如果 i 的初值大于 100，将使表达式 i <= 100 的值为 0，循环体也不执行。

②在循环体中一定要有使循环趋向结束的操作，以上循环体内的语句 i ++ 使 i 不断增 1，当 i > 100 时，循环结束。如果没有 i ++ ；这一语句，则 s 的值始终不变，进入死循环。

### 4.6.3　转移语句

程序中的语句通常是按顺序执行，但是需要改变程序的正常流向，可以使用转移语句，例如：如果循环条件需要中途退出循环时，可以考虑采用转移语句退出循环体。

**1. break 语句**

break 语句只能用在 switch 语句或者循环体语句中，其作用在循环体中遇见 break 语句，立即结束循环，跳到循环体外，执行循环结构后面的语句。break 语句的一般形式为

break;

break 语句可以使循环体语句有多个出口，使得编程更加灵活和方便。

例如：

```
for(r = 1; r < = 10; r ++)
  {
      area = PI * r * r ;
      if (area > 50)   break;
      printf("%f",area);
  }
```

程序中当 area > 50 时，使用 break 跳出循环。

**2. continue 语句**

continue 语句是一种中断语句，一般用在循环体中，其功能是结束本次循环，跳过循环体中下面尚未执行的语句，把程序流程转移到当前循环语句的下一个循环周期，并根据循环控制条件决定是否重复执行该循环体。

break 是跳出整个循环，continue 语句是跳出本次循环，也就是说出现在 continue 下面语句将不再执行，直接进入下一次循环。

continue 的一般格式为：continue;

例如：

```
main()
{
    int i;
    for(i = 1;i < = 20;i ++)
    {   if (i%5! =0) continue;
        printf("%d ",i);
```

```
        }
}
```

程序中，当变量 i 对 5 取模不等于 0 时，则为 i 不能整除 5，此时执行 continue 语句，程序将跳到 for 语句继续执行，只有模运算为 0 时，才能执行后面的 printf，输出能被 5 整除的数据。

## 4.7 本章小结

本章介绍了 C 语言的结构与用法，包括数据结构、数组和指针、函数以及程序设计语句的使用方法，其中 C 语言是 51 单片机软件编写的重点，通过本章的学习，对于初学单片机的读者掌握 C 语言流程控制语句设计的方法以及 C 语言一些基础知识，确保初学者具备基本的 C 语言软件编写能力。

# 第二部分　单片机基础案例实践篇

由电子元器件或者电子设备组成的能够完成一个特定应用功能的整体称为电子系统。任何电子系统都要通过电能的传输和转换，实现信号以及信息的产生、传输和处理，单片机是一种能够完成信息处理的电子器件。采用单片机对电子系统进行设计步骤如下：

1. 分析设计系统的功能和性能指标。
2. 提出系统方案，并分析系统工作原理。
3. 针对系统方案设计系统硬件原理图。
4. 针对系统硬件接口设计系统软件。
5. 结合硬件结构对系统软件进行调试。
6. 进一步完善和修改系统直至系统完成。

本书第二部分属于单片机基础实践知识内容，与第一部分单片机理论基础知识相结合，对于初学者有利于快速掌握单片机进行电子系统设计。单片机基础案例实践部分由11个章节内容构成。

第5章　单个LED点亮项目
第6章　花样流水灯闪烁项目
第7章　单片机独立按键控制项目
第8章　单片机外部中断控制项目
第9章　数码显示技术项目
第10章　单片机定时控制项目
第11章　单片机控制蜂鸣器项目
第12章　单片机串口通信项目
第13章　单片机实现4×4矩阵键盘控制项目
第14章　单片机实现字符型液晶显示项目
第15章　单片机实现步进电动机控制项目

# 第 5 章　单个 LED 点亮项目

## 5.1　项目需求

单片机发光二极管（LED）是基本的输出设备，通过 LED 有节奏地闪动显示系统的工作状态。LED 在电路中的作用一般用于电路状态提醒、广告装饰，也可以用于普通照明和城市夜景灯领域。本章介绍简单的 LED 闪动点亮项目。

## 5.2　项目工作原理分析

发光二极管（Light-Emitting Diode，LED）是一种能发光的半导体器件，它是半导体二极管的一种，可以把电能转化成光能。发光二极管与普通二极管一样由一个 PN 结组成，具有单向导电性。当给发光二极管加上正向电压后，从 P 区注入到 N 区的空穴和由 N 区注入到 P 区的电子，在 PN 结附近数微米内分别与 N 区的电子和 P 区的空穴复合，产生自发辐射的荧光，常用的是发红光、绿光或黄光的二极管如图 5-1 所示。



图 5-1　发光二极管实物图

发光二极管极性说明：LED 两根引线中较长的一根为正极，应接电源正极，较短一端为负极，电路符号如图 5-2 所示。有的发光二极管的两根引线一样长，但管壳上有一凸起的小舌，靠近小舌的引线是正极。一旦给 LED 通电，在它的正、负极加上合适的电压，LED 会亮起来。一般发光二极管工作电压在 1.6 ~ 2.8V 之间，而工作电流则一般在 2 ~ 30mA 之间。

采用单片机技术对 LED 进行闪亮控制，其电路结构相对简单，主要由单片机控制器、晶振电路、复位电路和输出设备 LED 构成，其电路原理结构图如图 5-3 所示。




图 5-2　发光二极管电路图符号

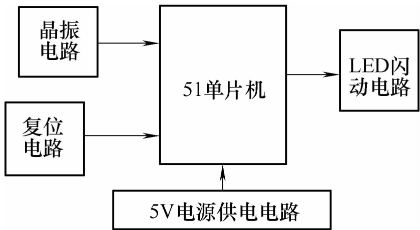



图 5-3　LED 闪动电路结构图

## 5.3　项目硬件电路设计

LED 闪动电路原理图如图 5-4 所示，通过转换电路将 220V 转换为 5V 电源，5V 电源电

路给 LED 闪动电路供电，LED2 是电源供电正常状态指示灯。单片机最小工作系统单元电路由晶振电路和复位电路构成，晶振电路采用内部振荡方式产生振荡时钟脉冲，通过单片机 18 引脚和 19 引脚外接晶体振荡器和微调电容相连，其中电容 C1 和 C2 电容容量为 30pF，晶振频率为 11.0592MHz。复位电路采用手动复位，通过单片机 9 引脚 RST 外接复位电路完成单片机上电复位功能，通过对引脚上连续保持两个机器周期以上的高电平，单片机就可以实现复位操作。LED 闪动电路由 LED1 和限流电阻构成，系统供电为 5V，若 LED1 上电电压是 2V，如果需要提高亮度，一般电流控制在 10mA 左右，此时电阻选择(5V－2V)/10mA＝300Ω，所以可以就近选择 330Ω，闪动电路通过单片机 P0.0 位输出高电平点亮 LED1 完成系统功能。

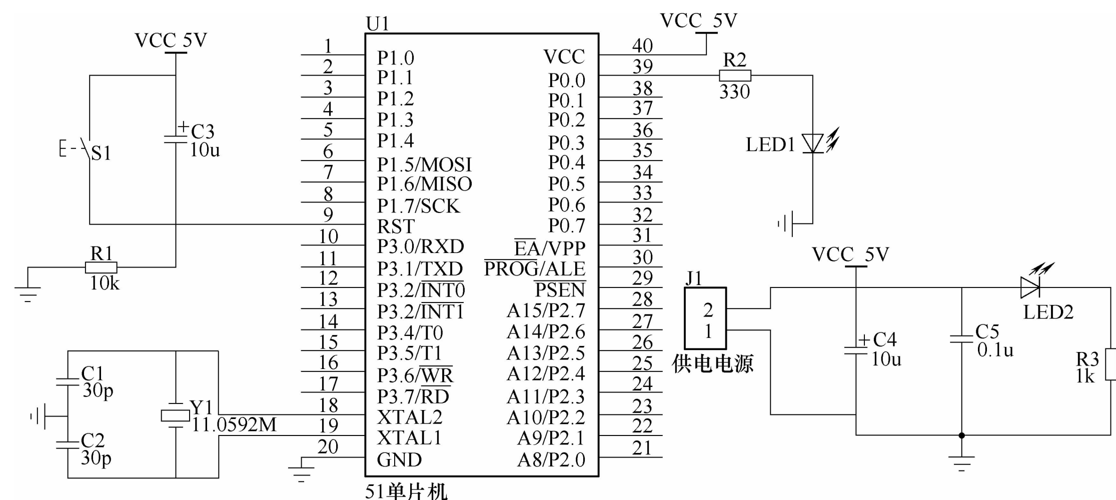



图 5-4　LED 闪动电路原理图

## 5.4　项目软件程序设计

系统设计的最终目的使得图 5-4 中的 LED1 处于闪动状态，即点亮 LED1 一段时间再熄灭 LED1，再点亮 LED1 再熄灭 LED1，如此循环下去。

软件设计过程中只要把 P0.0 置成高电平点亮 LED1，考虑到单片机的程序执行速度很快，若直接给 P0.0 置低电平熄灭 LED1，人眼无法看出 LED1 的亮灭状态，所以在点亮和熄灭 LED1 之间插入一个延迟函数，延迟时间足够长，因为延迟时间太短也是无法看到亮灭状态，所以一般闪动地延迟大约 300ms 左右，程序设计流程图如图 5-5 所示。

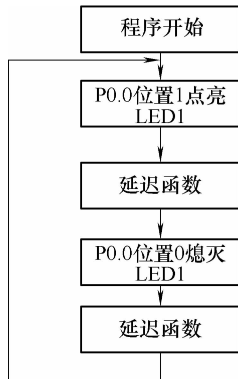



图 5-5　单个 LED 闪动程序流程图

依据图 5-5 程序设计思想实现单个 LED 的闪动状态，程序代码如下：

```
#include <reg51.h>
sbit LED1 = P0^0;            /* P0.0 位定义 */
```

```
void delay( )                    /* 定义延迟函数 */
{
     unsigned char m,n;
          for(m =0;m <250;m ++ )
               for(n =0;n <250;n ++ );
}                                /* 通过 m,n 的嵌套循环,实现延迟目的 */
void main( )
{
     while(1)                    /* 实现反复循环完成 LED1 灯的亮灭状态 */
     {
          LED1 =  1;             /*  点亮 LED1 */
          delay( );              /* 调用延迟函数 */
          LED1  =  0;            /*  熄灭 LED1 */
          delay( );              /* 调用延迟函数 */
     }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 LED1 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 5-6 所示。同时在创建工程路径 LED1 文件夹下生成一个扩展名为 . hex 文件，供下载软件将生成的 . hex 文件下载到单片机中。




图 5-6　程序成功编译结果图

## 5.5　系统调试结果总结

针对上述对系统硬件电路和软件程序的设计，通过在研展科技 YZ200 单片机开发板上调试单个 LED 闪动电路，将 Keil 编译完成后产生的 . hex 文件结合 2.4 节 ISP 程序烧写知识，将 . hex 文件烧写到 51 单片机中。对 LED 闪动电路通电可以看到单片机相连接的 P0.0 位的

LED1 循环亮灭，亮灭变化如图 5-7 所示，实现系统要求。

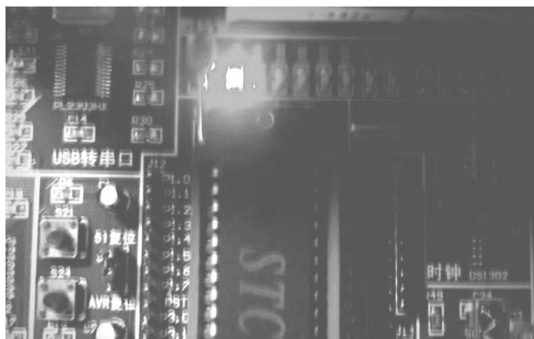



图 5-7　单个 LED 闪动电路调试成功图

# 第 6 章　花样流水灯闪烁项目

## 6.1　项目需求

第 5 章阐述了单个 LED 闪动系统的设计过程，本章针对日常生活中在商铺建筑物的棱角上安装流水灯，在夜间以 LED 流动发光的形式变换闪烁，有效地达到商铺广告促销的目的，对美化城市、点缀商业网点和引导消费起到积极的作用，而且还能在精神上给人美的享受。本章以 8 只 LED 流水灯为例，设计广告牌流水灯项目。

## 6.2　项目工作原理分析

本项目设计的花样流水灯闪烁项目是一个采用单片机控制 8 只发光二极管使其循环依次被点亮，流水灯闪烁状态见表 6-1，当 LED1 灯亮时，其他 7 只 LED 全部熄灭，当 LED2 灯亮时其他 7 只 LED 同样全部熄灭，以此循环下去实现 LED 流水闪烁。

**表 6-1　花样流水灯闪烁状态表**

| LED 序号 / 对应 I/O 口 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 | LED8 |
|---|---|---|---|---|---|---|---|---|
| P0.0 | | | | | | | | |
| P0.1 | | | | | | | | |
| P0.2 | | | | | | | | |
| P0.3 | | | | | | | | |
| P0.4 | | | | | | | | |
| P0.5 | | | | | | | | |
| P0.6 | | | | | | | | |
| P0.7 | | | | | | | | |

流水灯电路结构由单片机最小系统（包括单片机、晶振电路和复位电路）、电源电路和 8 盏流水灯电路构成，电路结构框图如图 6-1 所示。

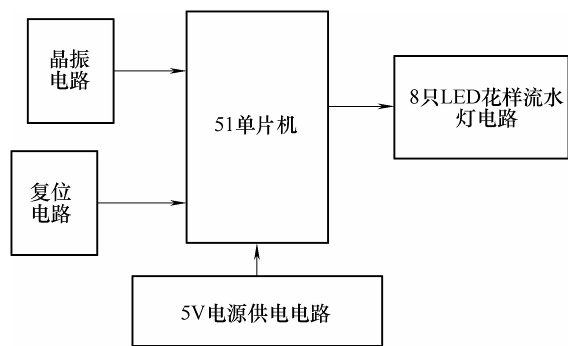



图 6-1　花样流水灯闪烁电路结构框图

## 6.3　项目硬件电路设计

花样流水灯进行电路设计时，其中电源供电电路、晶振电路和复位电路与第 5 章进行单个 LED 闪动项目设计时的原理是相同的，单片机 18 引脚和 19 引脚外接晶体振荡器和微调电容相连，电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz。复位电路采用手动复位，通过单片机 9 引脚 RST 外接复位电路完成单片机上电复位功能。8 只花样流水灯由 8 个 470Ω 的限流电阻和 8 个 LED 构成，限流电阻选择过大，LED 亮度不够，限流电阻选择太

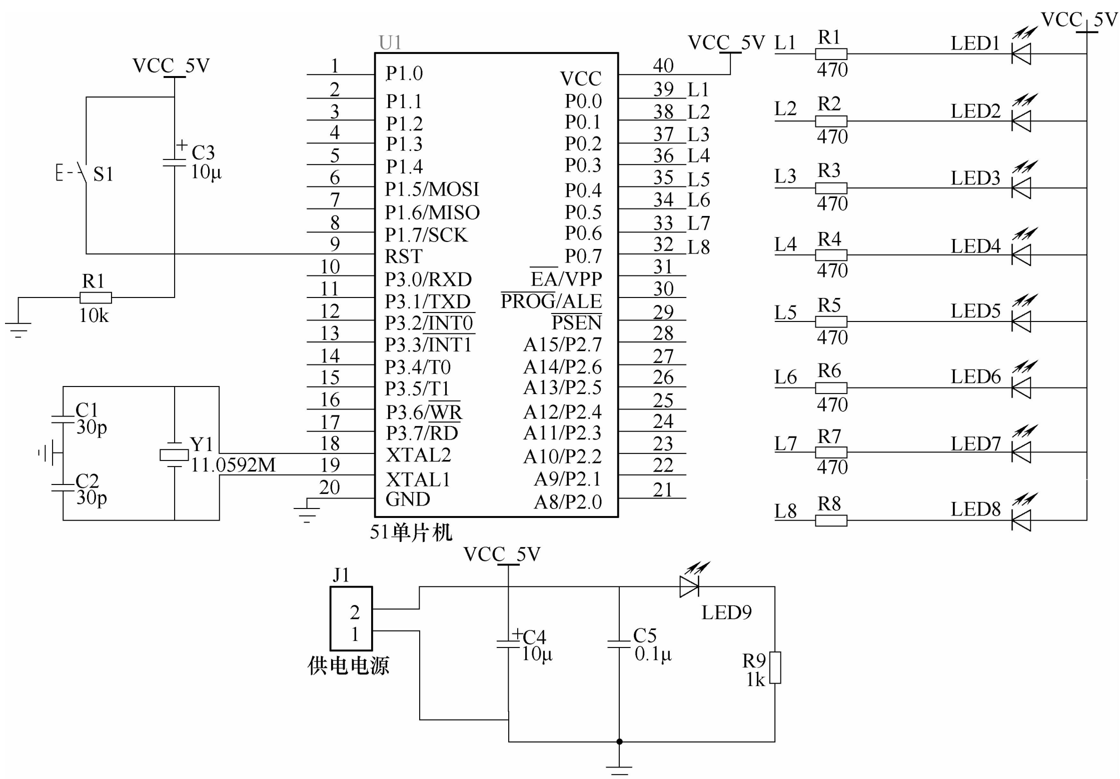



图 6-2　花样流水灯硬件电路原理图

小，一旦电流过大会烧坏 LED，考虑供电电压 5V，LED 上电电压 2V，流过 LED 的电流控制在 6 ~ 10mA，就近选择电阻为 470Ω。花样流水灯的硬件电路连接与第 5 章描述的单个 LED 闪动项目原理上相同，由于 8 只流水灯闪烁要求只是在软件设计上与单个 LED 闪动是不同的。

基于上述理论基础，参考图 6-1 的花样流水灯设计方案，设计出来的流水灯电路原理图如图 6-2 所示。

## 6.4 项目软件程序设计

从花样流水灯原理图 6-2 中可以看出，如果使其在 P0.0 口的 LED1 亮起来，那么只要置 P0.0 电平为低电平即可，如果使 P0.0 位的 LED1 熄灭，则 P1.0 电平置高电平。以此类推，P0.1 ~ P0.7 位连接的其他 7 只 LED 点亮和熄灭的方法与 LED1 点亮和熄灭方式相同。所以，为了实现流水灯功能，只要将发光二极管 LED1 ~ LED8 依次点亮、熄灭。流水灯流水样式见表 6-2，从表 6-2 可以看到通过逐个控制 P0 口的每个位来实现 LED1 ~ LED8 的亮灭，8 只 LED 便会一亮一暗做流水灯样式。但是如果按照 8 只 LED 亮灭顺序书写驱动软件，程序书写复杂。

**表 6-2 流水灯流水样式描述表**

| LED 点亮描述 | P0 口置数情况 |
|---|---|
| 点亮 LED1 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 01111111 |
| 点亮 LED2 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 10111111 |
| 点亮 LED3 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11011111 |
| 点亮 LED4 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11101111 |
| 点亮 LED5 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11110111 |
| 点亮 LED6 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11111011 |
| 点亮 LED7 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11111011 |
| 点亮 LED8 时其他 7 盏 LED 全部熄灭 | P0 口置数 = 11111110 |

尝试从优化程序结构简化程序角度书写流水灯软件程序，从表 6-2 中可以看出，8 只流水灯从左到右依次循环闪烁，将 8 只 LED 合成一个整体来考虑，利用循环移位指令，采用循环程序结构进行程序设计，在程序开始给 P0 口置数使得 P0.0 先为低电平，其他位为高电平，然后延时一段时间，再让这个数据向高位移动，然后再输出至 P0 口，从而实现“流水”效果。由于人眼的视觉暂留效应以及单片机执行每条指令的时间很短，所以控制发光二极管亮、灭的时间间隔应该延时一段时间，恰当的流水延迟时间设置非常重要，否则看不到“流水”效果。采用循环结构设计的流水灯右移循环闪烁程序流程图如图 6-3 所示。

```
#include <reg51.h>        //51 单片机寄存器定义的头文件
#include  <intrins.h>       //包含右移函数的头文件
void delayms(unsigned char ms) // 定义流水灯亮灭延迟时间
{
```

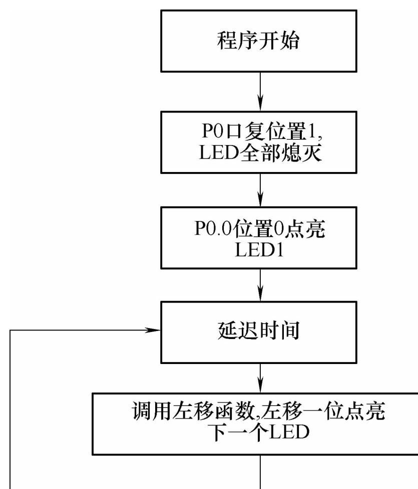



图 6-3　流水灯程序流程图

```
        unsigned char i;
        while(ms--)
        {
            for(i = 0; i < 120; i++);
        }
}
main()
{
        unsigned char LED;
        P0 =0xff;          // 给 8 只 LED 全部复位令其熄灭
        LED = 0xfe;   //点亮一只 LED,如 P0.0
        P0 = LED;
        while(1)      //引入循环程序结构
        {
            delayms(250);       //调用 LED 亮、灭延迟时间函数
            LED = _crol_(LED,1);      //循环右移 1 位,点亮下一只 LED
            P0 = LED;
        }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 LED2 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 6-4 所示。同时在创建工程路径 LED2 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

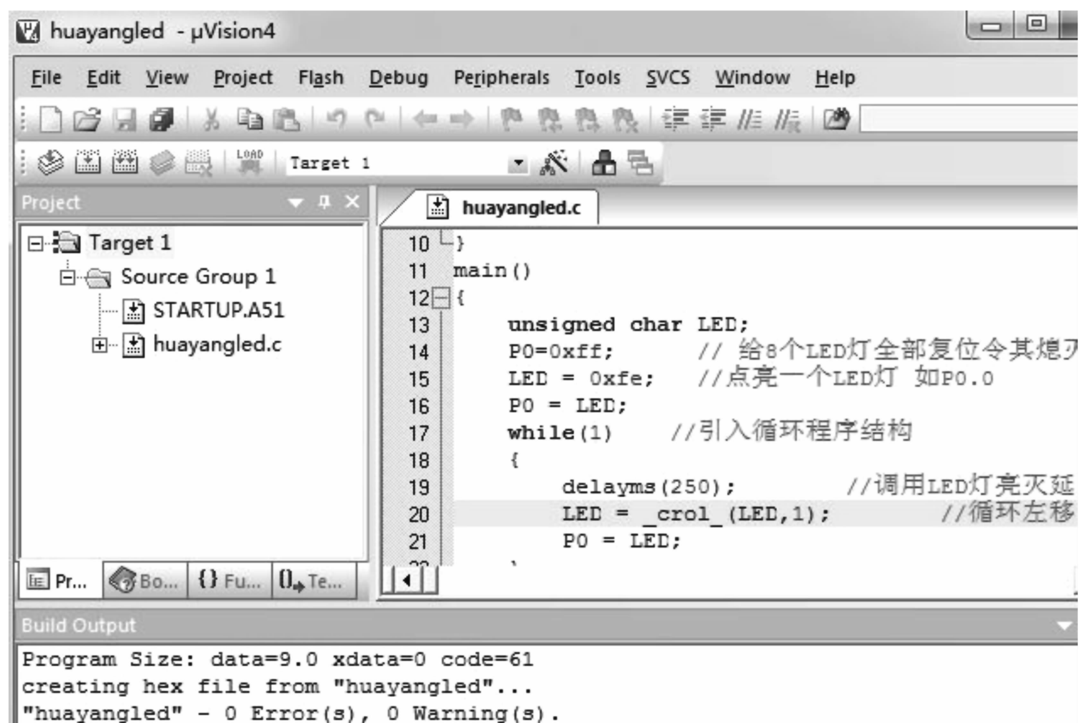



图 6-4　程序成功编译结果图

## 6.5　系统调试结果总结

针对上述对流水灯系统硬件电路和软件程序的设计，在研展科技 YZ200 单片机开发板上调试 8 只 LED 流水灯电路，将 6.4 节 Keil 编译完成后产生的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将 .hex 文件烧写到 51 单片机中。对 8 只 LED 流水灯电路通电可以看到单片机相连接的 P0 口的 8 只 LED 闪亮呈现流水状态，依次从左到右循环亮灭变化，实现流水灯系统要求。

# 第 7 章　单片机独立按键控制项目

## 7.1　项目需求

按键是电子系统设计时经常采用的一种数据输入元件，其实物图如图 7-1 所示，采用按键设计电路时一般有两种方式：一种方式为独立式按键，另外一种方式为矩阵式按键。本章主要介绍独立式按键的使用方法，电子系统设计时通过按动独立按键依据系统设计要求实现不同的功能。本章通过按动独立按键实现 LED 点亮项目，使得读者可以快速掌握独立按键的使用方法，设计要求为单片机 P3.0 ~ P3.3 连接独立按键，P0 口接 8 只 LED，每个按键控制两只 LED 的点亮与熄灭。

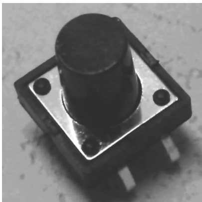


图 7-1　按键实物图

## 7.2　项目工作原理分析

从图 7-1 中可以看到按键有 4 个引脚，用万用表测试 4 个引脚的导通状况，测试结果显示一端两个引脚导通，另外一端两个引脚导通，而两端之间引脚不导通，以此构成两端口器件，一旦按下图 7-1 中黑色按钮，则独立的两端之间引脚导通，若黑色按钮没有按下，则独立的两端之间没有导通。在进行电路设计时一端引脚电平固定，另外一端引脚与单片机 I/O 口相连，通过对黑色按钮进行按下和不按下操作使得该引脚产生电平变化，单片机通过检测电平变化值完成对按键输入信息的获得，从而实现系统设计的各项功能。依据 7.1 节项目的需求，采用单片机对按键控制，电路结构由单片机最小系统（包括晶振电路和复位电路）、电源电路、按键电路和 LED 显示电路构成，电路结构图如图 7-2 所示。

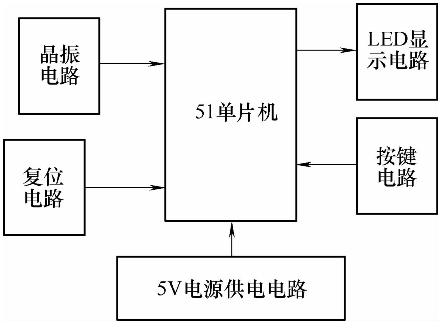



图 7-2　独立按键控制电路结构图

## 7.3　项目硬件电路设计

依据 7.2 节介绍按键控制电路由单片机最小系统、电源电路、按键电路和 LED 显示电路组成。其中电源电路 5V 供电、晶振电路设计时单片机 18 引脚和 19 引脚外接晶体振荡器和微调电容相连，电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz。复位电路采用手动复位，通过单片机 9 引脚 RST 外接复位电路完成单片机上电复位功能。P0 口与 8 只 LED 显示电路相连，其中 R1 ~ R8 为限流电阻防止电流过大烧坏 LED。独立按键电路通过单片机

的 P3.0～P3.3 与独立按键 S1～S4 相连，单片机通过不断监测是否有 S1～S4 按键按下，一旦有按键按下，则控制相应的 LED 点亮，其中 R9～R12 为上拉电阻。

其中独立按键识别流程如下：

①查询是否有按键按下？

②查询是哪个按键按下？

③执行按下按键相应的按键处理功能。

本项目 4 个独立按键控制 8 只 LED 的点亮规则如下：

按下 S1 按键控制 LED1 和 LED2 点亮。

按下 S2 按键控制 LED3 和 LED4 点亮。

按下 S3 按键控制 LED5 和 LED6 点亮。

按下 S4 按键控制 LED7 和 LED8 点亮。

依据上述理论基础，结合图 7-2 独立按键系统设计方案，设计出来的独立按键电路原理图如图 7-3 所示。



图 7-3 独立按键控制电路硬件原理图

## 7.4　项目软件程序设计

基于 7.3 节独立按键控制硬件电路如图 7-3 所示，可以看到单片机的 P3.0 ~ P3.3 外接上拉电阻，则 S1 ~ S4 无按键操作时，P3.0 ~ P3.3 端口保持高电平状态，当其 S1 按键按下时 S1 导通，按键直接与地相连，此时 P3.0 引脚为低电平，程序设计时只要检测到 P3.0 引脚电平为低电平，说明 S1 按键按下，则 P0.0 和 P0.1 置低电平，点亮 LED1 和 LED2；同理对于 S2 按键按下操作时，单片机不断监测 P3.1 引脚电平是否有变化，若有，则点亮 LED3 和 LED4；依此类推，单片机通过不断的监测 P3.0 ~ P3.3 电平的变化，点亮对应的 LED。

但是在编写独立按键控制 LED 亮的程序过程中需要考虑一个问题，按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开，因而在闭合及断开的瞬间均伴随有一连串的抖动，由于单片机执行程序速度很快，在按下按键这个时间里就会产生按键抖动。按键抖动会引起一次按键被误读多次，为确保单片机对按键的一次闭合仅作一次处理，必须去除按键抖动。在按键闭合稳定时读取按键的状态，一直判断到按键释放稳定后再处理其他语句。当按键数量操作比较多时，一般采用软件消抖的方法不断检测按键值，直到按键值稳定，实现方法为检测到按键输入为低电平后，延时一段时间（一般延时 5 ~ 10ms 恰好避开抖动期），再次检测，如果按键还是低电平，则认为有按键输入，再转去执行其他功能语句。独立按键控制电路程序设计流程图如图 7-4 所示。

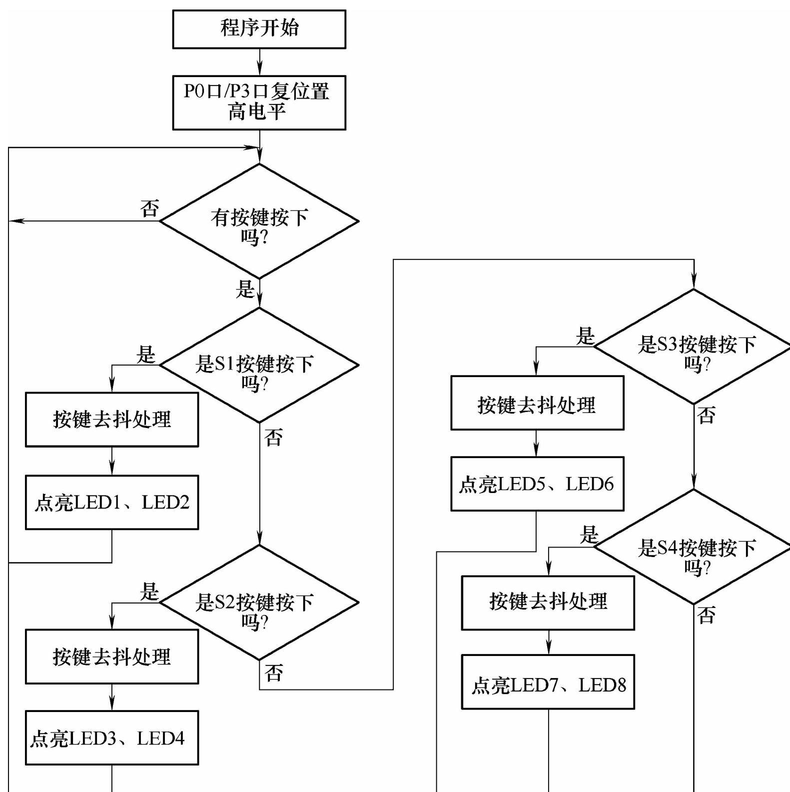



图 7-4　独立按键控制程序流程图

在独立按键控制程序流程图的基础上书写按键控制LED程序代码如下：

```
#include <reg51.h>          //51单片机寄存器定义的头文件
  sbit S1 = P3^0;          //定义S1按键与单片机P3.0位相连
  sbit S2 = P3^1;          //定义S2按键与单片机P3.1位相连
  sbit S3 = P3^2;          //定义S3按键与单片机P3.2位相连
  sbit S4 = P3^3;          //定义S4按键与单片机P3.3位相连
  void delay(void)        //定义延时函数用于按键去抖动
{
  unsigned char i,j;
  for(i=0;i<20;i++)
  for(j=0;j<250;j++);
}
  void main()
  {
      P3=0xff;             //对S1~S4按键复位
      P0=0xff;             //对8只LED复位
    while(1)               //是否有按键按下循环监测
    {
        if(S1==0)          //判断是否S1按键按下
        {
        delay();           //去抖动处理
        if (S1==0)         //确认S1按键按下
          {
          P0=0xfc;         //给P0口置数点亮LED1和LED2
        }
      }
    if(S2==0)              //判断是否S2按键按下
    {
    delay();               //去抖动处理
    if(S2==0)              //确认S2按键按下
      {
      P0=0xf3;             //给P0口置数点亮LED3和LED4
      }
    }
        if(S3==0)          //判断是否S3按键按下
        {
            delay();       //去抖动处理
            if(S3==0)//确认S3按键按下
            {
```

```
          P0 = 0xcf;        //给 P0 口置数点亮 LED5 和 LED6
          }
        }
          if(S4 ==0)        //判断是否 S4 按键按下
          {
            delay();        //去抖动处理
            if (S4 ==0)     //确认 S4 按键按下
            {
            P0 = 0x3f;      //给 P0 口置数点亮 LED7 和 LED8
            }
          }
      }
    }
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 button 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 7-5 所示。同时在创建工程路径 button 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

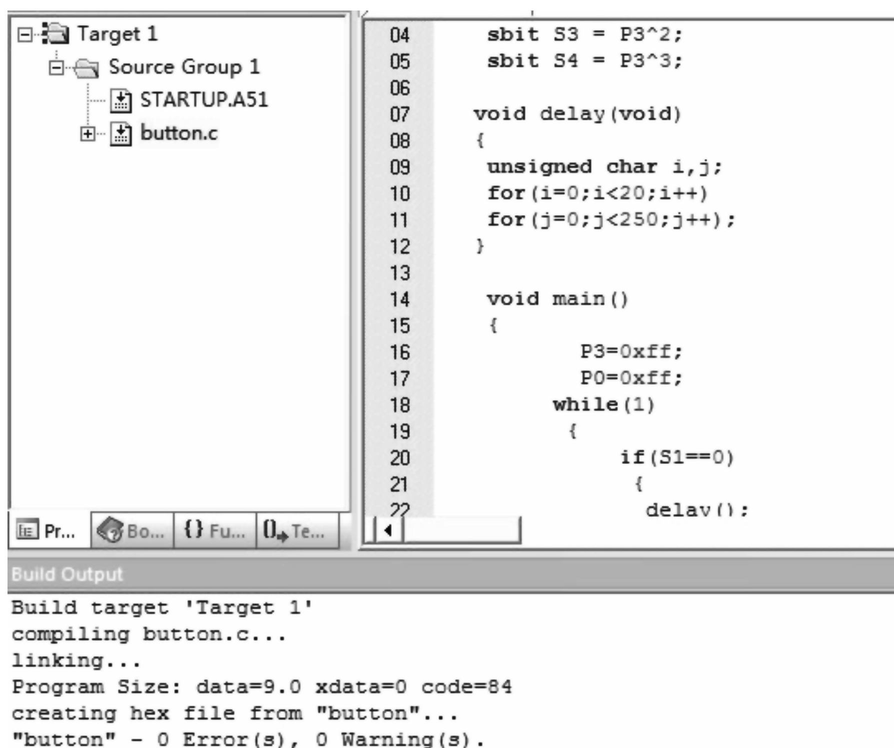



图 7-5　程序成功编译结果图

## 7.5 系统调试结果总结

针对上述对独立按键控制硬件电路和软件程序的设计，在研展科技 YZ200 单片机开发板上调试按键控制 LED 点亮电路，将 7.4 节 Keil 编译完成后产生的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将 .hex 文件烧写到 51 单片机中。通过在开发板上按动 S3 开关，可以看到 LED5 和 LED6 被点亮，实现独立按键控制要求。

# 第 8 章　单片机外部中断控制项目

## 8.1　项目需求

关于 51 单片机中断理论知识在第 3 章 3.3 节做了详细的讨论，读者在学习本章内容之前需要把 3.3 节内容预先复习。单片机在运行过程中，由于系统内部、外部等原因，需要单片机必须尽快终止正在运行的程序，转向相应的处理程序为其服务，等待处理完成后，再返回继续执行被终止的源程序，整个过程为中断控制过程。中断方式的引入大大提高了单片机工作效率以及处理问题的灵活性。为了能够使初学者快速掌握单片机中断知识，给出单片机中断控制项目需求为利用外部中断按键 S1 控制 P0 口 8 只 LED，中断时使 8 只 LED 闪烁 10 次。

## 8.2　项目工作原理分析

单片机有两个外部中断源分别为单片机 12 引脚 INT0 和 13 引脚 INT1，一般引脚外接按键，通过对按键操作产生外部中断输入信号，使得单片机响应中断执行中断服务程序，中断程序执行完成后，在返回到主程序之前发生中断的地方继续执行后面的语句。本项目主要针对 INT0 对单片机外部中断处理方式做详细说明，考虑 INT0 和 INT1 外部中断的使用方法原理相同，若读者采用 INT1 中断处理方式，可参考 INT0 中断处理方式。项目中若单片机检测到有外部中断 INT0 发生，依据第 3 章图 3-12 中断响应过程如下：

①INT0 中断源请求中断，则需要对 TCON 中的 IT0 和 IE0 进行设置，其中 IT0 位 =0 为电平触发，IT0 =1 为边沿触发。关于 TCON 特殊功能寄存器的使用详见第 3 章 3.3 节表 3-5。

②对①中请求的中断进行允许操作，则需要对中断允许寄存器 IE 中的 EX0 进行设置，其中 EX0 =1 允许外部中断 0，EX0 =0 禁止外部中断 0。

③在②基础上开启总中断开关 EA，其中 EA 是 IE 的中断总允许控制位，EA =1 开放总中断，EA =0 禁止总中断。关于 IE 中断允许寄存器的使用详见第 3 章 3.3 节表 3-7。

④经过①②③过程的操作，已经可以使用 INT0 外部中断处理方式，若系统只有一个中断处理，则忽略第④步只需要执行①②③过程即可，若系统有多个中断处理，则需要设定中断优先级，需要对中断优先级寄存器 IP 中的各位进行设置，PX0 =1 外中断 0 为高优先级，PX0 =0 外中断 0 为低优先级。关于 IP 中断优先级寄存器的使用详见第 3 章 3.3 节表 3-8。低中断优先级被高中断优先级中断，但是高中断优先级无法被低中断优先级中断；同级中断不能互相中断，CPU 按照硬件次

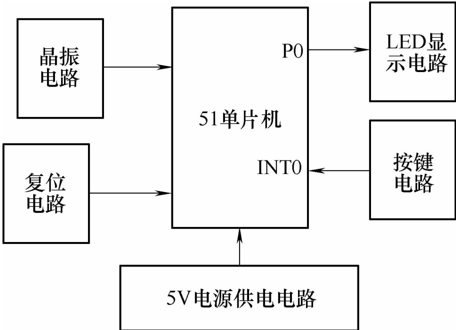


图 8-1　外部中断 0 控制电路结构框图

序决定优先权。同级优先权级别如图 3-11 所示。

经过上述 4 步运行，完成中断初始化操作。单片机不断对 INT0（即 P3.2 引脚）进行监测，一旦监测到中断源发生，执行中断初始化操作，打开中断执行中断服务任务。本项目中断服务任务每次中断发生，驱动 8 个 LED 闪烁 10 次。外部中断控制电路结构框图如图 8-1 所示，由单片机最小系统（包括晶振电路和复位电路）、电源电路、外部中断 0 发生电路（由按键电路触发）和 LED 显示电路构成。

## 8.3　项目硬件电路设计

依据图 8-1 所示，电路由单片机最小系统、电源电路、中断发生电路和 LED 显示电路组成。其中电源电路 5V 供电、晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz。复位电路采用手动复位。P0 口与 8 只 LED 相连构成驱动显示电路。外部中断 0 触发电路由独立按键构成，按键未按下前电平为高电平，一旦按键按下电平拉为低电平，单片机 P3.2 口有电平发生变化，触发中断，从而实现中断发生时驱动 8 只 LED 闪烁 10 次。外部中断 0 控制电路原理如图 8-2 所示。

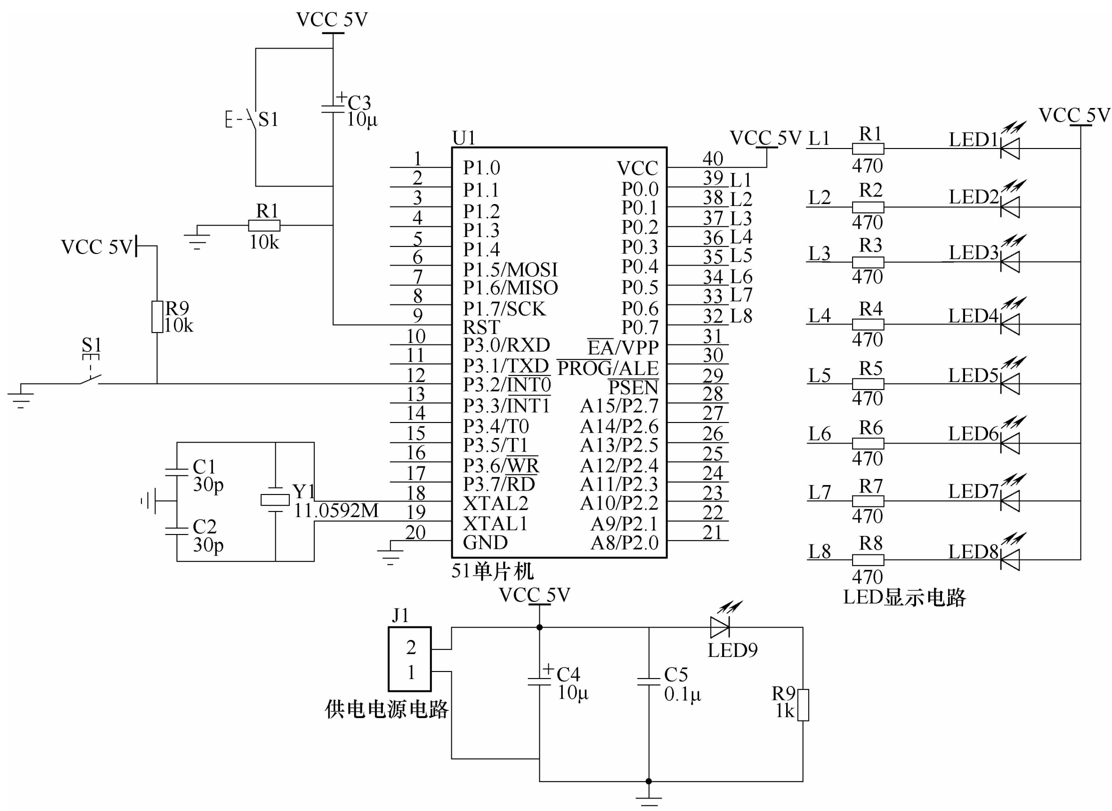



图 8-2　外部中断 0 控制电路原理图

## 8.4　项目软件程序设计

单片机 P0 口连接 8 只 LED，首先将 8 只 LED 复位，单片机 P3.2 口不断监测 INT0 是否有中断发生，一旦按键按下，通过对 TCON 中 IT0 设置选择中断触发模式，开启 EX0 和 EA，完成对 INT0 中断初始化操作。若单片机监测到中断发生，执行中断服务程序，考虑项目需求中断发生时驱动 8 只 LED 闪烁 10 次，通过构建循环语句实现 8 只 LED10 次闪烁，为了能够确保 LED 闪烁频率不要太快，通过编写延迟函数程序运行过程中调用延迟函数完成 LED "恰当" 的闪速。若 LED10 次轮询闪烁完成，依据中断原理，程序返回到主程序中断发生的地方，继续执行后面的语句。项目外中断 0 控制电路程序设计流程图如图 8-3 所示。

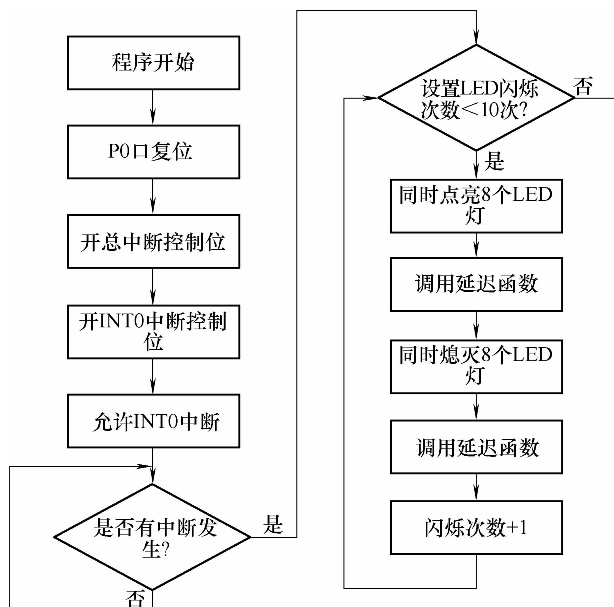



图 8-3　外中断 0 控制电路程序设计流程图

在图 8-3 程序流程图的基础上书写外中断 0 控制 LED 闪烁程序代码如下：

```
#include <reg51.h>
#define uchar unsigned char
void delay (unsigned int x)                    //定义延时函数
{
unsigned char i;
    while (x--)
     {
        for (i=0; i<125; i++)                  // 通过循环实现延迟功能
          {;}
     }
}
void intersvr0 (void) interrupt 0 using 1      //INT0 中断服务程序
{
uchar j;
    for (j=0; j<10; j++)                       // 8 只 LED 亮灭 10 次
        {
        P0 =0x00;                              //点亮 8 只 LED
```

```
        delay (500);                    // 调用延时函数保持 LED 亮状态
        P0 =0xff;                       //熄灭 8 只 LED
        delay (500);                    // 调用延时函数保持 LED 熄灭状态
    }
}
    void main (void)
{
P0 =0xff;                               //初始化 P0 口保持 LED 熄灭
EA =1;                                  //开启中断总控制位
IT0 =1;                                 //开启外部中断 0 控制位，采用边沿触发
EX0 =1;                                 //允许中断 0 中断
while (1);                              //判断是否有按键按下，若有按键按下执行中断
                                          服务程序
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 zhongduan 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 8-4 所示。同时在创建工程路径 zhongduan 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

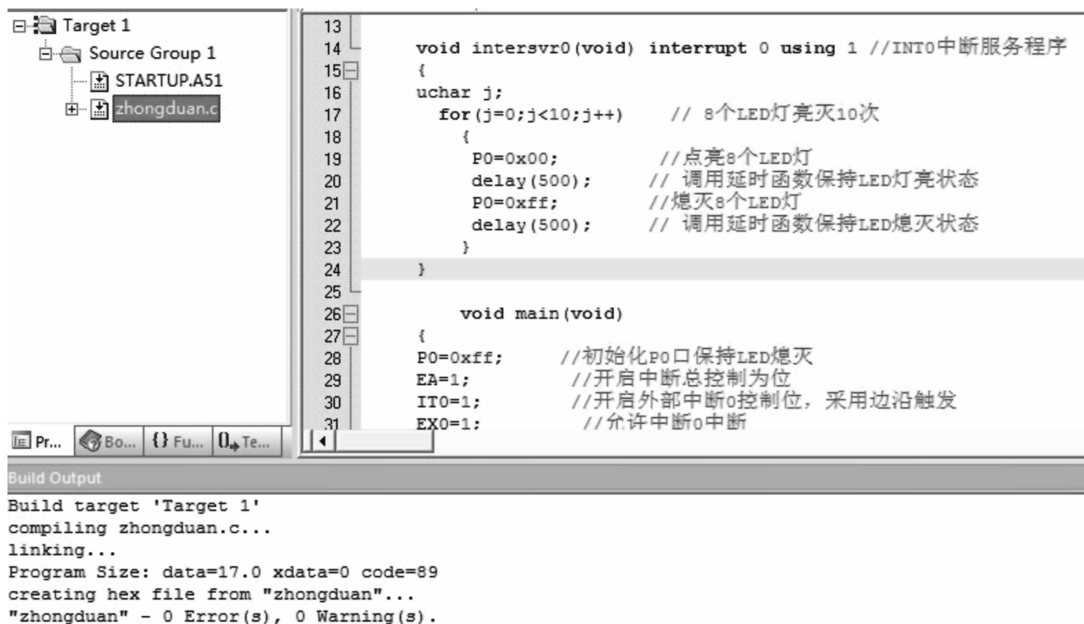



图 8-4 程序成功编译结果图

## 8.5 系统调试结果总结

针对上述对外中断 0 驱动 LED 闪烁硬件电路和软件程序的设计，在研展科技 YZ200 单

片机开发板上调试电路，将 8. 4 节 Keil 编译完成后产生的 . hex 文件结合 2. 4 节 ISP 程序烧写知识，将 . hex 文件烧写到 51 单片机中。通过在开发板上通电，按动 S1 开关触发中断，单片机监测中断发生，执行中断服务程序，即驱动 8 只 LED 闪烁 10 次，调试结果如图 8-5 所示，该结果是 8 只 LED 闪烁过程中拍摄结果，实现外中断 0 控制电路要求。

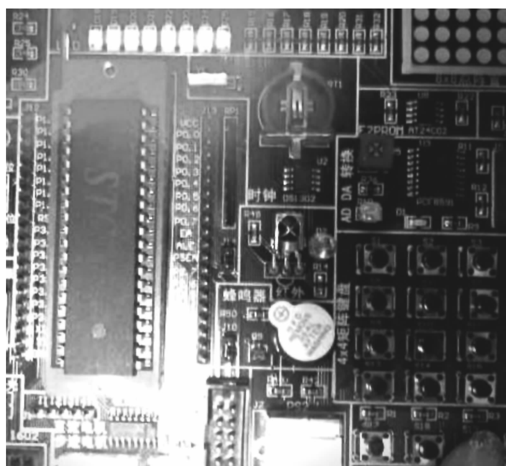



图 8-5　外中断 INT0 驱动 8 只 LED 闪烁结果

# 第 9 章　数码显示技术项目

## 9.1　项目需求

数码管（即 LED 数码显示器）是单片机与人机对话的重要输出设备。图 9-1 为数码管结构图，数码管由 8 个发光二极管组成，其中 7 个发光二极管组成 A 段、B 段、C 段、D 段、E 段、F 段和 G 段封装成“8”字形器件，另外一个发光二极管作为图 9-1 右下角小数点显示位。数码管在电子系统设计中主要应用于数据显示、定时控制等功能，其应用范围非常普遍。在单片机应用系统中，数码管显示方法有两种：①静态显示法；②动态扫描显示法。为了方便初学者快速掌握数码管的使用方法，我们采用数码管静态显示和数码管动态显示方法实现对数码管的控制。设计要求是在 4 位集成共阳型数码管上每隔一定时间循环显示“2”“4”“6”“8”数字。

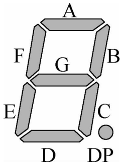
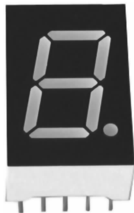


图 9-1　数码管结构图

## 9.2　项目工作原理分析

图 9-2 为 1 位数码管实物图，由 8 只 LED 构成“8”字符号和一个小数点，其引脚图如图 9-3 所示，1 位数码管有 10 个引脚，引脚说明如下，其中 a ~ h 段为段码端，因为 1 位数码管只显示 1 位“8”字，所以公共端 1 脚和 3 脚是导通的为位码端，注意 1 位数码管只有 1 个位码。

| | |
|---|---|
| 1 引脚—e 段 | 2 引脚—d 段 |
| 3 引脚—公共端 | 4 引脚—c 段 |
| 5 引脚—h 段 | 6 引脚—b 段 |
| 7 引脚—a 段 | 8 引脚—公共端 |
| 9 引脚—f 段 | 10 引脚—g 段 |


图 9-2　1 位数码管实物图

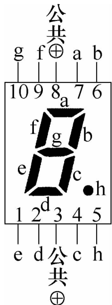


图 9-3　1 位数码管引脚图

数码管在使用过程中有时候需要显示多位数据，一般按照显示多少个“8”可分为 1 位、2 位、3 位、4 位、5 位、6 位、7 位等数码管，其显示原理是相同的，根据项目要求显示“2”“4”“6”“8”共 4 位数据则需要 4 位数码管，考虑优化电路硬件结构，选择集成 4 个 1 位数码管构成 4 位数码管实物图如图 9-4 所示，其集成 4 位数码管引脚如图 9-5 所示。其中 a ~ h 段为段码端，集成 4 位数码管有 4 个位码分别为 com1、com2、com3 和 com4 4 个公共端。

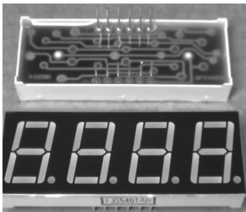


图 9-4　集成 4 位数码管

| | |
|---|---|
| 1 引脚—e 段 | 2 引脚—d 段 |
| 3 引脚—h 端 | 4 引脚—c 段 |
| 5 引脚—g 段 | 6 引脚—com4 端 |
| 7 引脚—b 段 | 8 引脚—com3 端 |
| 9 引脚—com2 端 | 10 引脚—f 段 |
| 11 引脚—a 段 | 12 引脚—com1 端 |

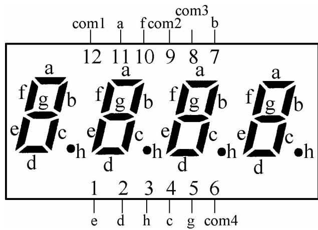



图 9-5　集成 4 位数码管引脚图

鉴于数码管内部驱动方式不同分成两种类型：①共阳型数码管；②共阴型数码管。

共阳型数码管：8 只 LED 的正极全部连接在一起组成公共端，其内部 LED 连接方式如图 9-6 所示，使用时公共端接 5V，段码端接低电平驱动 LED 导通点亮。

共阴型数码管：8 只 LED 的负极全部连接在一起组成公共端，其内部 LED 的连接方式如图 9-7 所示，使用时公共端接地，段码端接高电平驱动 LED 导通点亮。

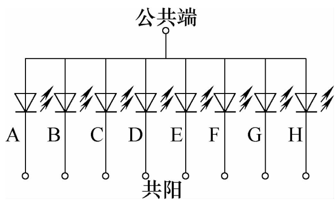



图 9-6　共阳型数码管

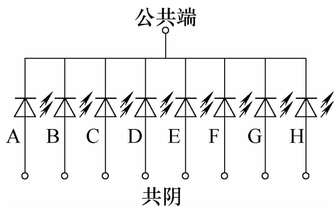



图 9-7　共阴型数码管

从图 9-6 和图 9-7 中看到，共阳型数码管公共端接高电平，段码端接低电平，相应的字段被点亮。为了显示数字或者符号，如显示“3”，即点亮数码管 a，b，c，d，g 段，使其对应段码接低电平，即“10110000”，同时给公共端置高电平，此时显示“3”数字。共阴型数码管使用方法与共阳型数码管相反，共阴型数码管公共端接低电平，段码端接高电平，相应的字段被点亮。若显示“3”数字，即点亮数码管 a，b，c，d，g 段，使其对应段码接高电平，即“01001111”，同时给公共端置低电平，此时显示“3”数字。

基于数码管显示原理，依据 9.1 节项目需求，采用单片机驱动数码管显示数据，电路结构由单片机最小系统（包括晶振电路和复位电路）、电源电路和数码显示电路构成，电路结构图如图 9-8 所示。

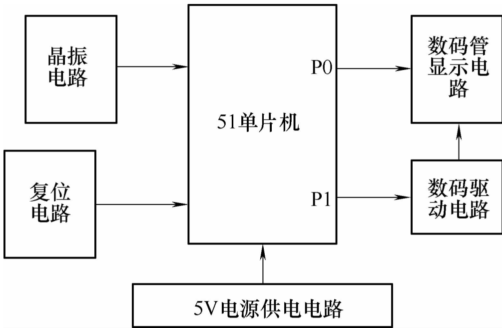



图 9-8　数码显示电路结构框图

## 9.3 项目硬件电路设计

依据图 9-8 所示，电路由单片机主芯片、电源电路、晶振电路、复位电路、数码驱动电路和数码显示电路组成。其中电源电路 5V 供电；晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz；复位电路采用手动复位。数码驱动电路采用 8 位三态锁存器 74HC353 芯片用于放大电流驱动数码显示，74HC573 芯片工作时使能信号端 1 引脚 OC 端置低电平，11 引脚 C 端置高电平，输入端 1D ~ 4D 与单片机 P1.0 ~ P1.3 位相连，输出端 1Q ~ 4Q 与集成 4 位数码管位码端 com1 ~ com4 相连。集成 4 位数码管段码端 a 段 ~ h 段与单片机 P0.0 ~ P0.7 位相连，选择集成 4 位数码管为共阳型数码管，通过设定数码管段码端（即单片机 P0.0 ~ P0.7 位）为低电平，设定数码管位码端（即单片机 P1.0 ~ P1.3 位）为高电平，就可以根据设计要求在数码管上实现需要显示的数据。设计的数码显示电路硬件电路原理图如图 9-9 所示。

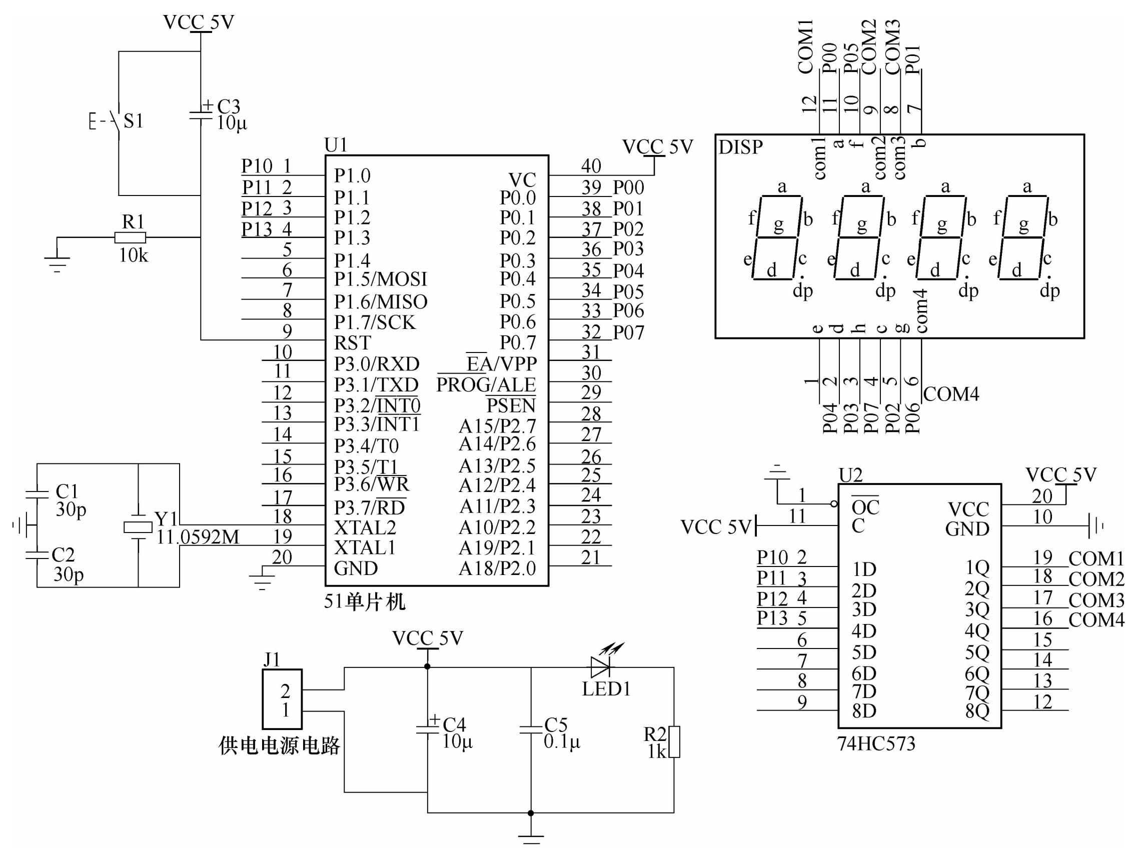



图 9-9 数码显示电路硬件电路图

# 9.4　项目软件程序设计

单片机对数码管显示电路进行软件设计时一般有两种方法：①静态显示法；②动态扫描显示法，无论采用哪种方法实现数码管显示功能，其系统设计的硬件电路结构不发生改变，只是程序书写流程上发生变化。我们将从数码管静态和动态显示两个方面进行程序设计。

## 9.4.1　数码管静态显示

静态显示是指集成 4 位数码管各位码端（即 com1 ~ com4 公共端）全部占用具有锁存功能的输出口线，单片机把显示字形对应的段码给集成 4 位数码管 a 段 ~ h 段赋值，即显示出设计所需要显示的数字或者符号。

静态显示法单片机不会经常扫描数码管各位码端，节省程序运行时间，但是由于程序运行时占用全部的公共端口，所以该方法一般用在数码管数目较少的应用系统中。

采用静态显示方式进行程序设计时，考虑显示电路选取的是共阳数码管，若实现数码管显示“2”“4”“6”“8”，则需要通过对数码管位码和段码进行设置方能实现。如观察“4”是如何显示出来的，首先点亮数码管 b、c、f、g 段，对应到单片机的 I/O 口，将 P0.1、P0.2、P0.5 和 P0.6 端口置“0”，公共端 com1 ~ com4 置“1”，实现集成 4 位数码管全部显示“4”。表 9-1 列出显示“4”数字段码对应关系。

**表 9-1　共阳数码管显示数字“4”段码表**

| 段名称 | h | g | f | e | d | c | b | a | 对应段码 |
|---|---|---|---|---|---|---|---|---|---|
| 对应引脚 | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 | |
| 数字 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0x99 |

进行数码管显示电路设计时，选取数码管器件可以共阴也可以共阳，表 9-2 列出共阴和共阳数码管 0 ~ F 十六个数字的段码表，段码表可以通用，方便读者在进行数码管软件编程时使用。

**表 9-2　数码管字形段码表**

| 显示字形 | 段码(共阳) | 段码(共阴) | 显示字形 | 段码(共阳) | 段码(共阴) |
|---|---|---|---|---|---|
| 0 | 0xc0 | 0x3f | 8 | 0x80 | 0x7f |
| 1 | 0xf9 | 0x06 | 9 | 0x90 | 0x6f |
| 2 | 0xa4 | 0x5b | a | 0x88 | 0x77 |
| 3 | 0xb0 | 0x4f | b | 0x83 | 0x7c |
| 4 | 0x99 | 0x66 | c | 0xc6 | 0x39 |
| 5 | 0x92 | 0x6d | d | 0xa1 | 0x5e |
| 6 | 0x82 | 0x7d | e | 0x86 | 0x79 |
| 7 | 0xf8 | 0x07 | f | 0x8e | 0x71 |

书写程序时，通过查表 9-2 查找段码，调用段码把该段码数据送到 P0 口实现段码显示。

程序设计上通过循环结构实现“2”“4”“6”“8”数字的轮询显示。数码管静态显示程序流程图如图 9-10 所示。

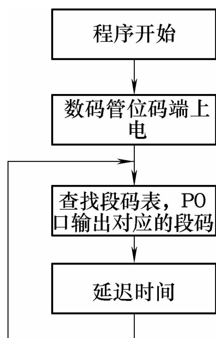



图 9-10　数码管静态显示程序流程图

在图 9-10 的基础上书写静态数码显示程序代码如下：

```
#include   <reg51.h>   //51 单片机寄存器定义头文件
unsigned char code Table[ ] = {0xA4,0x99,0x82,0x80};
        //  定义共阳型数码管显示的 2、4、6、8 对应的段码
    sbit com1 = P1^0;  //定义集成 4 位数码管的位码端 com1 与 P1.0 口相连
    sbit com2 = P1^1;  //定义集成 4 位数码管的位码端 com2 与 P1.1 口相连
    sbit com3 = P1^2;  //定义集成 4 位数码管的位码端 com3 与 P1.2 口相连
    sbit com4 = P1^3;  //定义集成 4 位数码管的位码端 com4 与 P1.3 口相连
    void delay(unsigned int x)          //定义延时函数
    {
    unsigned char i;
        while(x--)
        {
            for(i=0;i<125;i++)   //通过循环实现延迟功能
            {;}
        }
    }
    void main()
    {
    unsigned char i=0;
    com1 =1;  //设置集成 4 位共阳数码管位码端 com1 上电
    com2 =1;  //设置集成 4 位共阳数码管位码端 com1 上电
    com3 =1;  //设置集成 4 位共阳数码管位码端 com1 上电
    com4 =1;  //设置集成 4 位共阳数码管位码端 com1 上电
    while(1)
        {
```

```
            for(i=0;i<4;i++) //通过循环结构实现数码管轮询显示 2、4、6、8
            {
            P0=Table[i];
    //通过 P0 口设置集成 4 位数码管的段码，对数组 Table 的调用点亮数码管
            delay(500);  //调用延迟函数确保数码管保持点亮状态
            }
        }
    }
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 disp 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 9-11 所示。同时在创建工程路径 disp 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

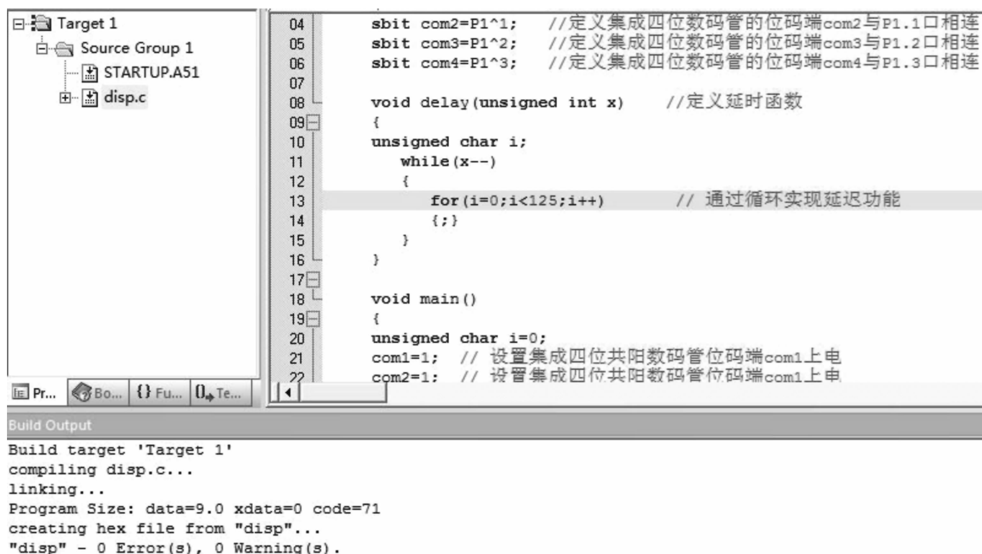



图 9-11　数码静态显示程序编译成功界面

### 9.4.2　数码管动态显示

数码动态扫描显示方法是段码用来控制显示的字形，位码用来控制选择第几位数码管工作。通过对段码和位码的操作，可以一位一位轮流点亮各个数码管对应的字形，实现动态扫描。在轮流点亮过程中，每位数码管点亮的时间是极为短暂（时间大约 1 ~ 5ms）的，促使人眼看数字的点亮是连续的，而不是一位一位的显示。原因在于数码管内部的 LED 具有余辉特性以及人眼视觉的惰性，虽然数码管分时显示，但是只要恰当选取扫描频率，人眼会觉得数码管是连续稳定的显示。

书写数码管动态扫描程序时，通过查找段码表把段码送 P0 口，控制位码点亮相应位的数码管，调用延迟时间函数，之后熄灭该位数码管，依此循环实现下一位数码管字形的显示。延迟时间函数书写时要求极为短暂，使得人眼会觉得数码管是连续稳定的显示。数码管动态扫描程序流程图如图 9-12 所示。

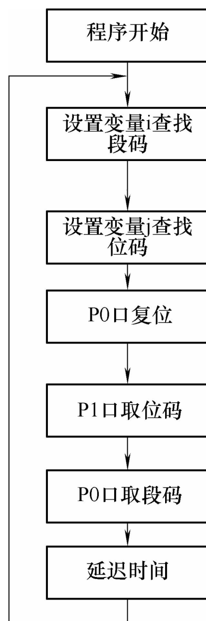



图 9-12 数码管动态扫描显示程序流程图

在图 9-12 基础上书写数码动态扫描显示程序代码如下:

```
#include   <reg51.h>   //51 单片机寄存器定义头文件
unsigned char code duanma[] = {0xA4,0x99,0x82,0x80};
//定义共阳型数码管显示的 2、4、6、8 对应的段码
unsigned char code weima[] = {0x01,0x02,0x04,0x08};
//定义共阳型数码管位码,点亮相应位的数码管
void delay(unsigned int t)     //定义延时函数
{
    while(--t);
}
void main()
{
unsigned char i,j;
for(i=0;i<4;i++)     //循环查找段码
    for(j=0;j<4;j++)   //循环查找位码
    {
    P0=0xff;   //清空数据,防止交替阴影
    P1=weima[j];   //取位码
    P0=duanma[i]; //取段码
    delay(1500);     //扫描间隙延时,数字 2、4、6、8 在集成 4 位数码管上轮询闪烁
    }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习,建立工程 disp1 文件,并将上述代码

在 Keil 环境下进行编译，程序成功编译结果如图 9-13 所示。同时在创建工程路径 disp1 文件夹下生成一个扩展名为 . hex 文件，供下载软件将生成的 . hex 文件下载到单片机中。




图 9-13　数码动态扫描显示程序编译成功界面

## 9.5　系统调试结果总结

### 9.5.1　数码管静态显示调试结果

针对数码显示硬件电路和静态显示软件程序，在研展科技 YZ200 单片机开发板上调试数码管静态显示电路，将 9. 4. 1 节 Keil 编译完成后产生的 . hex 文件结合 2. 4 节 ISP 程序烧写知识，将 . hex 文件烧写到 51 单片机中。开发板通电，可以看到集成 4 位数码管循环显示“2”“4”“6”“8”数字，考虑拍照截图原因，数码管静态显示数字“6”时调试结果如图 9-14 所示，实现数码静态显示结果。

### 9.5.2　数码管动态扫描显示调试结果

针对数码显示硬件电路和动态扫描显示软件程序，在研展科技 YZ200 单片机开发板上调试数码管动态扫描显示电路，将 9. 4. 2 节 Keil 编译完成后产生的 . hex 文件结合 2. 4 节 ISP 程序烧写知识，将 . hex 文件烧写到 51 单片机中。开发板通电，可以看到集成 4 位数码管动态扫描显示“2”“4”“6”“8”数字，数码管动态显示数字“8”时调试结果如图 9-15 所示，实现数码动态扫描显示结果。



图 9-14　数码静态显示数字“6”结果图



图 9-15　数码动态扫描显示数字“8”结果图

# 第 10 章　单片机定时控制项目

## 10.1　项目需求

关于 51 单片机定时理论知识在第 3 章 3.4 节做了详细的讨论，读者在学习本章内容之前需要把 3.4 节内容预先复习。定时器是单片机机器周期的计数器，每个机器周期包含晶体振荡器的 12 个振荡周期，每个机器周期定时器加 1，其频率为晶振频率的 1/12。如果晶振频率 $f_{osc}$ 为 12MHz，则一个机器周期为 $12/f_{osc}=2\mu s$。若选择计数器工作方式时，计数脉冲来自单片机引脚 P3.4（定时/计数器 T0）或者引脚 P3.5（定时/计数器 T1），一旦检测到 P3.4 和 P3.5 引脚上电平发生跳变，则计数器加 1。51 单片机定时/计数器可用于定时控制、延时、计数和检测操作，其应用十分广泛。为了能够使初学者快速掌握单片机定时知识，给出单片机定时控制项目需求：选取单片机晶振频率 $f_{osc}$ 为 12MHz，单片机 P0.1 脚接 1 只 LED 发光二极管，通过采用定时器 T0 工作方式 1 控制 LED 亮 65ms 和灭 65ms，循环闪烁。

## 10.2　项目工作原理分析

51 单片机定时/计数器内部结构及其工作原理具体详见第 3 章 3.4 节，内部设置两个定时/计数器 T0 和 T1，具有计数器和定时器两种工作方式以及 4 种工作模式，把定时/计数初始值写入 TH 和 TL 中控制定时/计数长度；通过对 TCON 的有关位置数和清零启动定时/计数器工作和停止。在使用 51 单片机定时/计数器时，关于定时/计数器初始化一般遵循 4 步如下：

①确定工作方式，设置 TMOD。

②计算定时/计数初值，并将其装载到 TH 和 TL。

③定时采用中断控制，则设置 IE 开启定时/计数器中断。

④启动定时/计数器，采用软件启动设置 TCON 中的 TR1 或者 TR0；若采用外部中断引脚电平启动，则需要给引脚外加启动电平。

一旦实现启动定时/计数器功能，则定时/计数器按照预先设定的工作方式和计算好的初值开始定时或者计数。通过对 TMOD 中的 M1M0 进行设置选择相应的 4 种工作方式，考虑定时器 T0 工作方式 3 时，占用定时器 T1 的 TR1 和 TF0，此时定时器 T1 只能工作于串行通信波特率发生器模式，所以本章主要阐述工作方式 0、工作方式 1 和工作方式 2，其对应计算定时/计数值做如下总结：

（1）工作方式 0　定时器装载定时/计数初值为 TH 高 8 位和 TL 低 5 位。

①作为计数工作

计数值 $=2^{13}-$ 计数初值

②作为定时器工作

定时时间 $=(2^{13}-$ 定时初值$)\times$ 机器周期

（2）工作方式 1　定时器装载定时/计数初值为 TH 高 8 位和 TL 低 8 位。

①作为计数器工作

计数值 $=2^{16}-$ 计数初值

②作为定时器工作

定时时间 $=(2^{16}-$ 定时初值$)\times$ 机器周期

（3）工作方式 2　定时器装载定时/计数初值为 TH 高 8 位和 TL 低 8 位。

①作为计数器工作

计数值 $=2^{8}-$ 计数初值

②作为定时器工作

定时时间 $=(2^{8}-$ 定时初值$)\times$ 机器周期

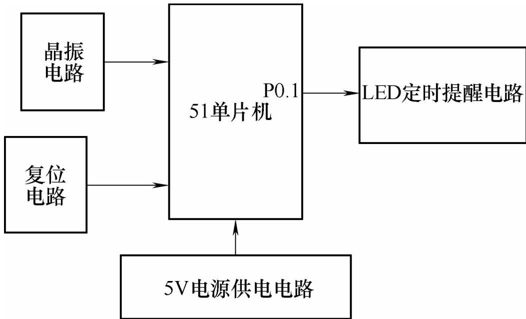



图 10-1　定时器 T0 控制 LED 闪烁电路结构框图

本项目中要求采用定时器 T0 处于工作方式 1 控制 P0. 1 脚连接的 LED 循环闪烁亮 65ms，灭 65ms。采用定时方式控制 LED 闪烁的系统电路结构框图如图 10-1 所示，由单片机最小系统（包括晶振电路和复位电路）、电源电路、LED 定时提醒电路构成。

## 10. 3　项目硬件电路设计

依据图 10-1 所示，电路由单片机最小系统、电源电路、LED 定时提醒电路组成。其中电源电路 5V 供电、晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 12MHz。复位电路采用手动复位。P0. 1 引脚外接 LED 构成定时提醒电路，采用单片机内部定时器 T0 将定时初值装载到 TH0 和 TL0 中实现定时功能，单片机 P0. 1 口依据 T0 定时功能驱动 LED 循环闪烁亮 65ms，灭 65ms。设计的电路原理图如图 10-2 所示。

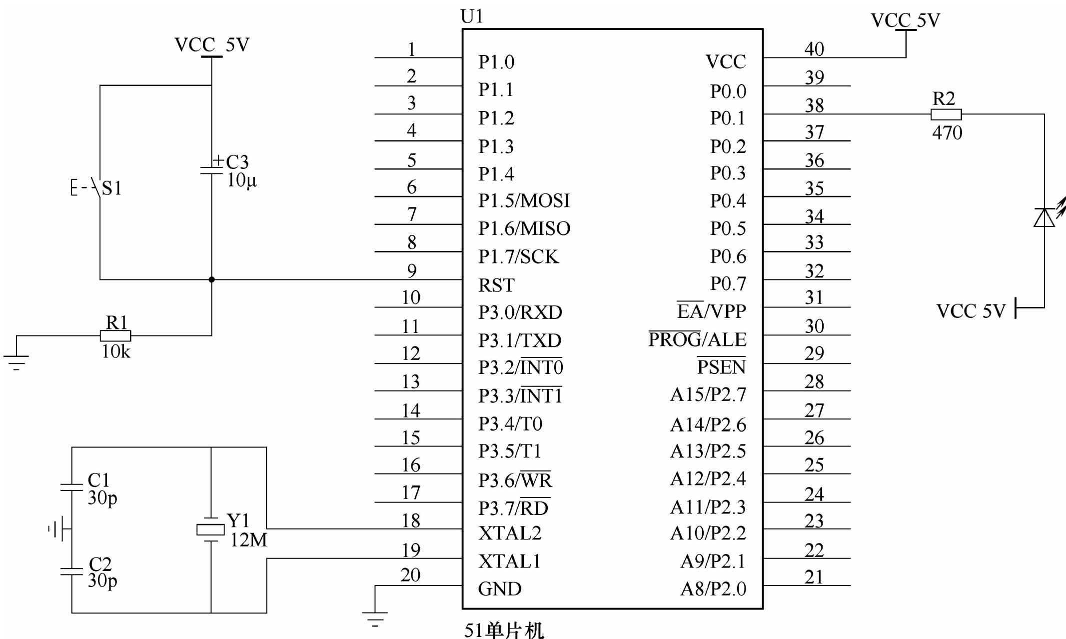



图 10-2　定时器 T0 控制 LED 闪烁电路原理图

## 10.4 项目软件程序设计

依据项目需求驱动 LED 亮 65ms，灭 65ms，只是需要在 P0.1 端以 65ms 为周期交替输出高、低电平实现。设计硬件电路晶振为 12MHz，则机器周期为

$$机器周期 = 12/晶振频率 = 12/12MHz = 1\mu s$$

采用定时器 T0 工作方式 1，设置 M1M0 = 01；为实现定时功能，使得 C/T = 0。由定时器控制寄存器 TCON 中的 TR0 = 1 启动定时器 T0，选择工作方式 1 时 T0 为 16 位定时器。定时时间 65ms，采用工作方式 1 作为 T0 定时初值计算公式可以得到

T0 初值 $= 2^{16} - 65000\mu s/1\mu s = 65536 - 65000 = 536 = 218H$，则高八位为 02H，放入 TH0 中，低八位为 18H，放入 TL0 中，即 TH0 = 02H，TL0 = 18H。对 T0 定时器控制 LED 闪烁书写软件一般有两种方式：①采用中断方式；②采用查询方式。采用中断方式书写程序时，T0 运行工作后开始从初值加 1 计数，至最高位产生溢出时，申请中断，单片机响应中断执行中断服务程序。采用查询方式书写程序时，利用循环结构判断 T0 定时器最高位溢出标志 TF0 是否为 1，若 TF0 = 1 则 T0 计数器溢出开始下一轮计数。

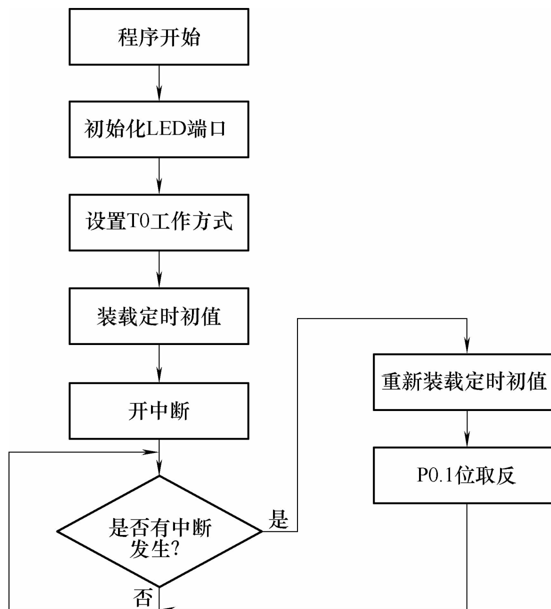



图 10-3　中断方式 T0 控制 LED 程序流程图

用定时器 T0 方式 1 编写程序，即采用中断方式设计的程序流程图如图 10-3 所示。

在图 10-3 程序流程图的基础上书写中断实现 T0 控制 LED 闪烁程序代码如下：

```
#include <reg51.h>                //包含头文件
sbit LED = P0^1;                  //端口定义
void main(void)                   //主程序入口
{
    P0 = 0xff;                    //初始化 LED 端口
    TMOD = 0x01;                  //选择 T0 方式 1 计时
    TH0 = 0x02;                   //装载 T0 的高 4 位初值
    TL0 = 0x18;                   //装载 T0 的低 4 位初值
    EA = 1;                       //开启总中断
    ET0 = 1;                      //允许 T0 中断
    TR0 = 1;                      //启动 T0 运行
    while(1);                     //T0 定时溢出时请求中断
}
```

```
void time0(void)   interrupt 1 using 1//定时中断服务程序入口
{
    TH0 =0x02;                    //重新装载 T0 的高 8 位初值
    TL0 =0x18;                    //重新装载 T0 的低 8 位初值
    LED = !LED;                   //P0.1 位取反
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 time 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 10-4 所示。同时在创建工程路径 time 文件夹下生成一个扩展名为 . hex 文件，供下载软件将生成的 . hex 文件下载到单片机中。

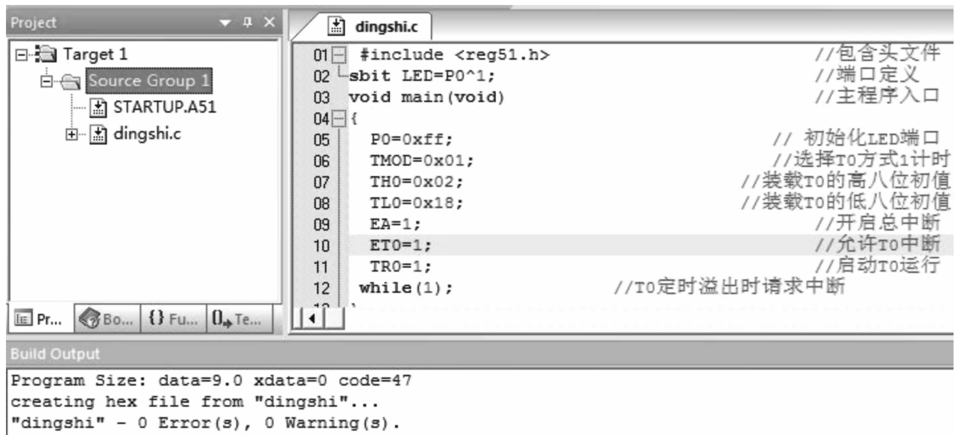



图 10-4　程序成功编译结果图

用定时器 T0 方式 1 编写程序，采用查询方式程序流程图如图 10-5 所示。

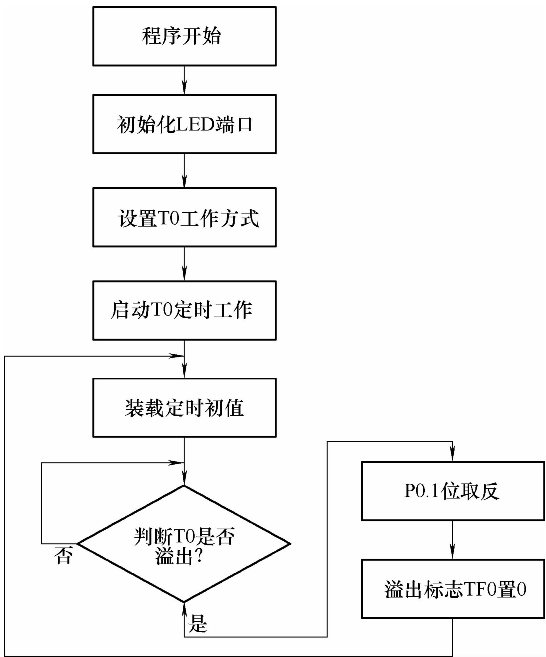



图 10-5　查询方式 T0 控制 LED 程序流程图

在图 10-5 程序流程图的基础上书写查询实现 T0 控制 LED 闪烁程序代码如下：

```
#include  <reg51.h>                    //包含头文件
sbit LED = P0^1;                       //端口定义
void main(void)                        //主程序入口
{
  P0 = 0xff;                           // 初始化 LED 端口
  TMOD = 0x01;                         //选择 T0 方式 1 计时
  TR0 = 1;                             //启动 T0 运行
  while(1)
  {
    TH0 = 0x02;                        //装载 T0 的高 8 位初值
    TL0 = 0x18;                        //装载 T0 的低 8 位初值
    do{ }while(! TF0);                 //查询等待 TF0 置位
    LED = !LED;                        //P0.1 位取反
    TF0 = 0;                           //软件对 T0 溢出标志清“0”
  }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 time1 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 10-6 所示。同时在创建工程路径 time1 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

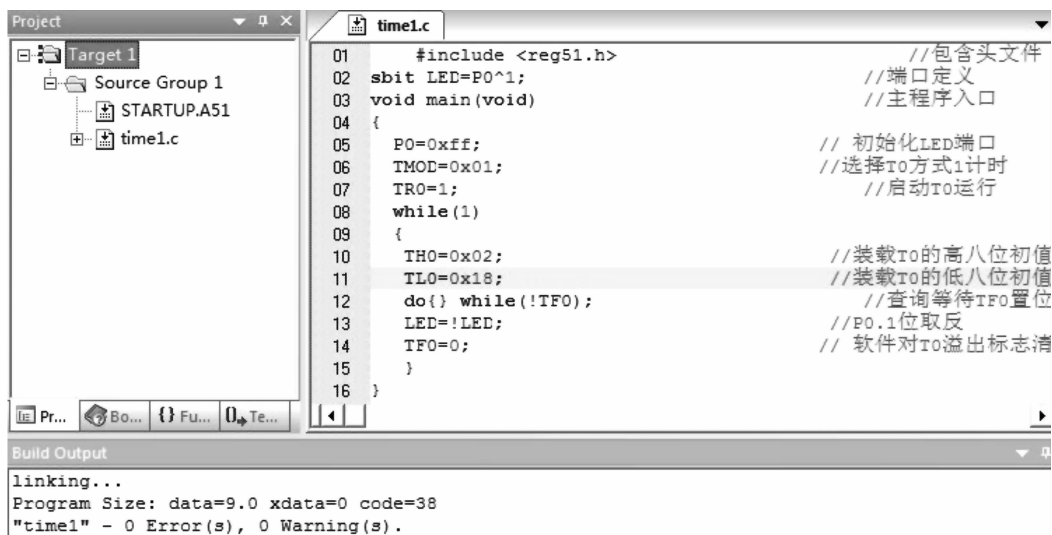



图 10-6　程序成功编译结果图

## 10.5　系统调试结果总结

针对定时器 T0 控制 LED 定时闪烁的硬件电路和软件程序的设计，在研展科技 YZ200 单

片机开发板上调试电路，将 10.4 节中采用中断方式和查询方式控制定时器 T0 编译生成的两个 .hex 文件，结合 2.4 节 ISP 程序烧写知识，分别将采用两种方式编程生成的 .hex 文件烧写到 51 单片机中。通过在开发板上通电，分别采用两种编程方法调试电路，看到的调试结果是一致的，结果显示单片机 P0.1 端连接的 LED 65ms 点亮、65ms 熄灭，循环反复，实现系统要求。

# 第 11 章　单片机控制蜂鸣器项目

## 11.1　项目需求

蜂鸣器是一种一体化结构的电子讯响器，采用直流电压供电，广泛应用于报警器、电子玩具、汽车电子设备、定时器等电子产品中作发声器件。在单片机应用设计上，很多方案都会用到蜂鸣器，一般使用蜂鸣器作提示或报警，如按键按下、开始工作、工作结束或是故障等。蜂鸣器形状如图 11-1 所示，有自己的固有频率，可以加不同频率方波编制一些简单的声音。为了方便初学者掌握蜂鸣器的使用方法，通过项目实践快速上手。项目要求：通过单片机控制蜂鸣器循环发出警报声，作为电路报警提示电路。




图 11-1　蜂鸣器实物图

## 11.2　项目工作原理分析

蜂鸣器按照极性要求加上合适的直流电压，可以发出固有频率的声音，使用简单。由于蜂鸣器是直流电压驱动，不需要利用交流信号，只需对驱动口输出驱动电平并通过晶体管放大驱动电流使蜂鸣器发出声音。采用单片机 P3. 5 脚定时翻转电平产生驱动波形对蜂鸣器进行驱动，利用定时器做定时，通过定时翻转电平产生符合蜂鸣器要求频率的波形用来驱动蜂鸣器发声。蜂鸣器控制电路结构框图如图 11-2 所示，由单片机最小系统（包括晶振电路和复位电路）、电源电路和蜂鸣器报警提示电路构成。

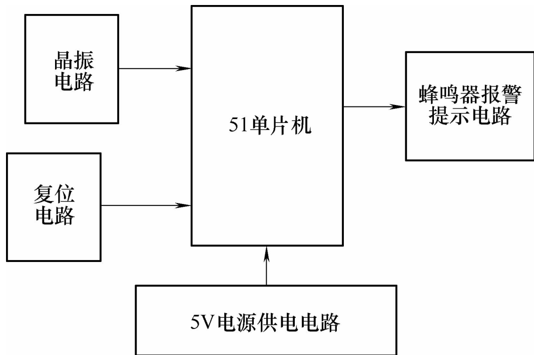



图 11-2　蜂鸣器控制电路结构框图

## 11.3　项目硬件电路设计

依据图 11-2 进行项目硬件电路设计，电路由单片机最小系统、电源电路和蜂鸣器报警

提示电路组成。其中电源电路 5V 供电、晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz。复位电路采用手动复位。蜂鸣器通过晶体管驱动电路与单片机的 P3.5 引脚相连。

蜂鸣器是一个感性负载，直接固定直流电压，电路导通即发出固定频率的声音。不建议用单片机 I/O 口直接对蜂鸣器操作，一般在单片机 I/O 口与蜂鸣器之间加个驱动晶体管（如 PNP 型晶体管 9012），主要用来电流放大，当 P3.5 脚输出低电平时，晶体管导通，形成回路，蜂鸣器工作。设计的硬件电路原理图如图 11-3 所示。

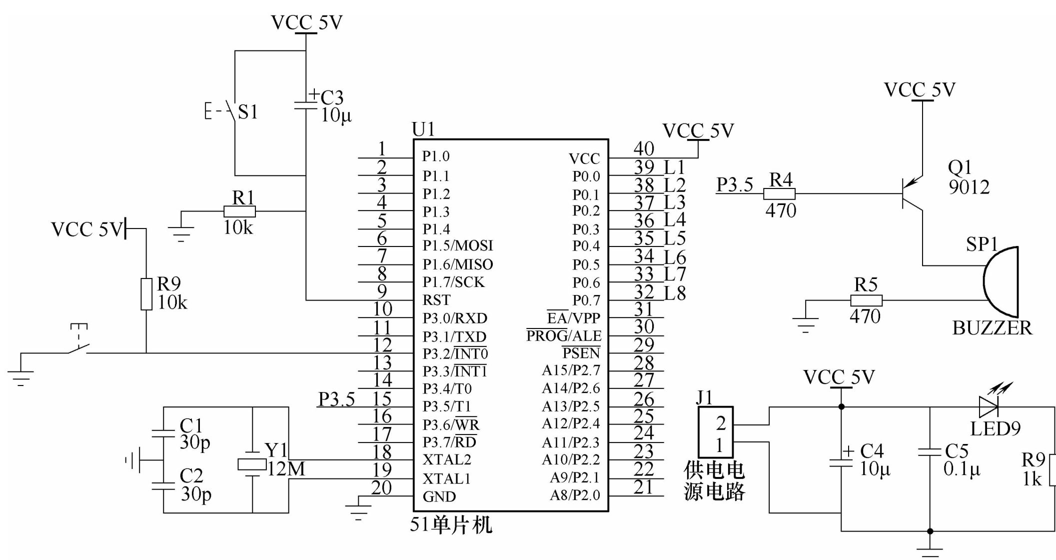



图 11-3　蜂鸣器控制硬件电路原理图

## 11.4　项目软件程序设计

从 11.3 硬件电路图中可以看到 P3.5 端通过晶体管 Q1 驱动蜂鸣器 BUZZER 发声，使用 P3.5 脚定时翻转电平驱动蜂鸣器发声。程序设计上，使用 T0 定时工作方式 1，先给予 T0 定时时间初值（如定时 65ms），T0 被允许计数后，T0 从初值开始加 1 计数，至最高位产生溢出时 TF1 = 1，T0 定时请求中断，则定时器 T0 转入执行中断服务子程序，此时对 P3.5 引脚的电平进行翻转一次，蜂鸣器鸣叫。中断子程序中将更新的定时初值重新装载到 T0 定时器中，即将初值高八位和低八位数据写入 TH0 和 TL0 中，再次等待计数置最高位产生溢出，若 TF0 = 1 计数器溢出请求中断，则 T0 定时器再次进入中断服务子程序驱动蜂鸣器鸣叫，然后再次更新定时初值重新装载到 T0 定时器中，等待下一次中断响应，依此反复循环实现蜂鸣器报警提示功能。蜂鸣器控制电路程序流程如图 11-4 所示。

在图 11-4 程序流程图的基础上书写蜂鸣器报警控制电路程序代码如下：

```
#include <reg51.h>          //定义单片机寄存器头文件
#include <intrins.h>        //定义函数头文件
```

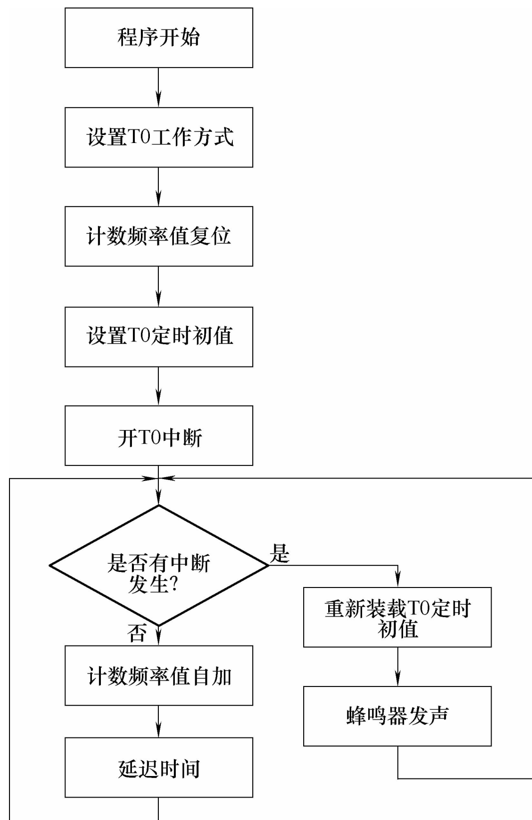



图 11-4 蜂鸣器控制电路程序流程图

```
  sbit fengming = P3^5;              //定义蜂鸣器端口
  unsigned char frq;
  void delay(unsigned char ms);      //定义延迟函数声明
main()
{
     TMOD = 0x01;                    //定义 T0 定时器工作方式 1
     frq = 0x00;
     TH0 = 0x00;                     //给定初值
     TL0 = 0xff;
     TR0 = 1;                        //启动 T0 工作运行
     IE = 0x82;                      //允许 T0 中断
     EA = 1;                         //开启总中断
     while(1)
     {
       frq++;                        //延迟时间,累积频率值
       delay(10);
```

```
    }
}
void timer0() interrupt 1 using 1
{
  TH0 = 0xfe;                    //装载定时初值高 8 位
  TL0 = frq;                     //低 8 位值在 main() 函数中不断累加
  fengming = ~fengming;          //蜂鸣器端口电平取反
}
void delay(unsigned char ms)     //定义延迟函数
{
  unsigned char j;
  while(ms--)
  {
    for(j=0;j<200;j++);
  }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 sound 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 11-5 所示。同时在创建工程路径 sound 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。




图 11-5　程序成功编译结果图

## 11.5 系统调试结果总结

针对蜂鸣器报警提醒硬件电路和软件程序的设计，在研展科技 YZ200 单片机开发板上调试蜂鸣器报警电路，将 11.4 节 Keil 编译完成后产生的 . hex 文件结合 2.4 节 ISP 程序烧写知识，将 . hex 文件烧写到 51 单片机中。通过给开发板上电，可以听到蜂鸣器消防报警提示声音，实现项目要求。

# 第 12 章　单片机串口通信项目

## 12. 1　项目需求

串口（Serial Interface）是计算机上一种通用设备通信的协议，也是仪器仪表设备通用的通信协议，串行接口线实物图如图 12-1 所示。串口通信时，数据一位一位地顺序传送，其特点是通信线路简单，只要一对传输线就可以实现双向通信，降低了成本，特别适用于远距离通信，但传送速度较慢。单片机进行串行通信时通过单片机引脚 RXD（P3. 0 串行数据接收端）和引脚 TXD（P3. 1 串行数据发送端）与外围电路进行通信，通信过程只需向发送缓冲器写入数据，从接收缓冲器读出数据即可。如果在串行口的输入、输出引脚加上电平转换器（一般使用 MAX232），将串口转换为标准的 RS232 接口实现单片机与 PC 通信。为了能够使初学者快速掌握单片机串行通信知识，给出项目需求：单片机接收 PC 发送的数据，然后将接收到的数据传送到 P0 口，并传回给 PC；一旦按下 S2 时，则单片机发送字符“BEST”到 PC 端。本项目是一个结合 LED、按键、电平转换电路和单片机控制应用相结合的项目，属于相对综合性的案例。



图 12-1　串行接口线实物图

## 12. 2　项目工作原理分析

从第 3 章 3. 5 节可知，51 单片机有一个全双工的串行通讯口，所以单片机和计算机之间可以方便地进行串口通信。51 单片机串行口内部结构及其工作原理具体详见第 3 章 3. 5 节，内部有两个缓冲器 SBUF（serial Buffer），一个作发送缓冲器，另一个作接收缓冲器。串行通信波特率时钟必须从内部定时器 T1 获得。串行口工作方式有 4 种，具体描述参见 3. 5. 3 节。在使用 51 单片机串行口之前中首先对串行口初始化，包括设置产生波特率定时器 1、串行口控制和中断控制。具体步骤如下：

①设置定时器 T1 工作方式。

②计算波特率并将初值装载到 T1 的 TH1 和 TL1 中。

③启动定时器 T1。

④设定串行口工作方式。

⑤若采用中断处理则开启中断。

通常情况下，使用单片机串口时选用晶振相对固定，本项目电路晶振选择 12MHz，与 PC 通信时，串口选用波特率相对固定，常用波特率及相应设置见表 3-15，本项目选择串行口工作方式 1 波特率 1200 波特，其定时器 T1 的初值，E6H 装载到 TH1 和 TL1 中设置波特率。单片机与 PC 通信时，由于单片机输出 TTL 电平，而 PC 为 RS232 电平，则需要经过 MAX232 电平转换电路实现通信。项目实现串行通信电路结构框图如图 12-2 所示，由单片机最小系统（包括晶振电路和复位电路）、电源电路、按键电路、LED 显示电路以及电平转换 MAX232 电路构成。

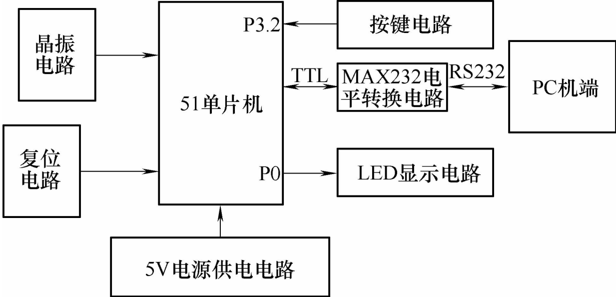



图 12-2　串行通信电路结构框图

## 12.3　项目硬件电路设计

依据图 12-2 所示，电路由单片机最小系统、电源电路、按键电路、LED 显示电路以及电平转换 MAX232 电路组成。其中电源电路 5V 供电、晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 12MHz，复位电路采用手动复位。按键电路通过单片机的 P3. 2 引脚与 S2 独立按键相连，单片机监测到 S2 按键按下，则发送字符“BEST”到 PC 端。LED 显示电路与单片机的 P0 口相连，PC 端发送数据给单片机，单片机将接收到的数据传送给 P0 口驱动 LED 点亮。PC 与单片机通信时要满足一定的条件，PC 的串口是 RS232 电平的，单片机的串口是 TTL 电平的，两者之间必须有一个电平转换电路，采用专用芯片 MAX232 进行转换，MAX232 外部引脚如图 12-3 所示。

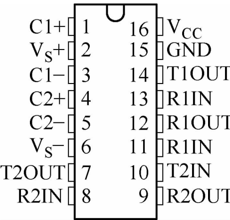



图 12-3　MAX232 引脚图

从图 12-3 中按照引脚使用功能可以分成三部分：

第一部分是电荷泵电路：由 1、2、3、4、5、6 脚和 4 只 0.1μF 电容构成。功能是产生 12V 和 -12V 两个电源，提供给 RS232 串口电平需要。

第二部分是数据转换通道：由 7、8、9、10、11、12、13、14 脚构成两个数据通道。其中 13 脚（R1IN）、12 脚（R1OUT）、11 脚（T1IN）、14 脚（T1OUT）为第一数据通道。8 脚（R2IN）、9 脚（R2OUT）、10 脚（T2IN）、7 脚（T2OUT）为第二数据通道。TTL/CMOS 数据从 T1IN、T2IN 输入转换成 RS232 数据从 T1OUT、T2OUT 送到计算机 DP9 插头，DP9 插头的 RS232 数据从 R1IN、R2IN 输入转换成 TTL/CMOS 数据后从 R1OUT、R2OUT 输出。

第三部分是芯片供电：15 脚接地、16 脚 VCC 接 5V 电源。

MAX232 电平转换外接电路如图 12-4 所示。

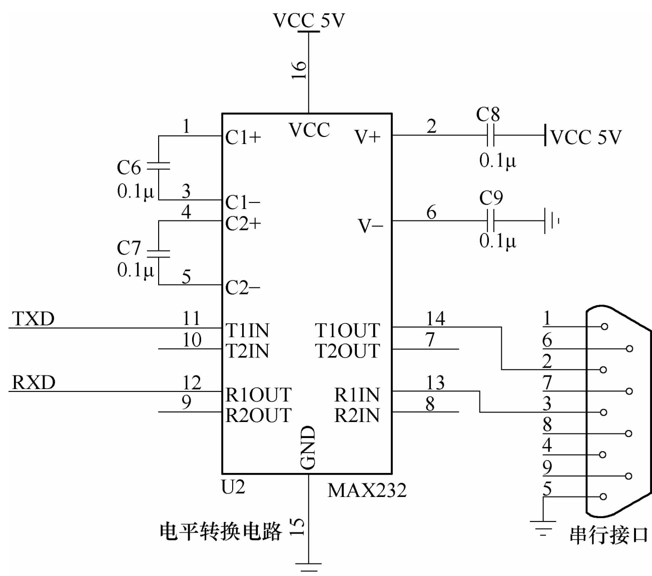



图 12-4　MAX232 电平转换外接电路

依据上述阐述，结合图 12-2 电路原理结构图，对串行通信电路原理图进行设计，则电路图如图 12-5 所示。

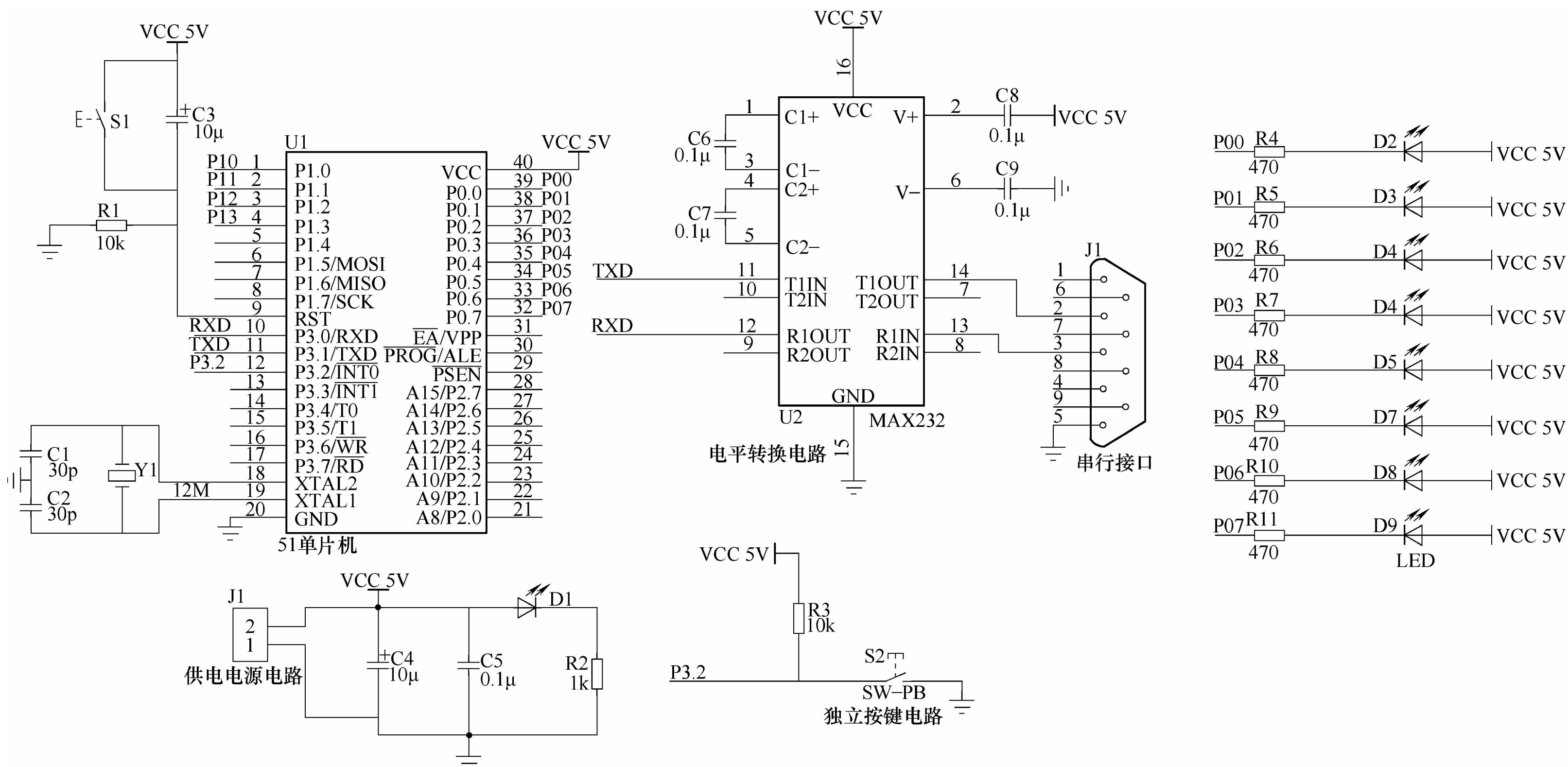



图12-5　串行通信硬件电路原理图

## 12.4　项目软件程序设计

依据项目需求设计硬件电路晶振为 12MHz，则机器周期为

$$\text{机器周期} = 12/\text{晶振频率} = 12/12\text{MHz} = 1\mu s$$

采用定时器 T1 工作方式 1，通过设置 M1 M0 = 10，用于 8 位自动重载模式产生波特率，选定波特率为 1200 波特，参考表 3-15 将 T1 初值 E6H 分别装载到高 8 位 TH1 和 TL1 中。设置串行口工作方式 SM0 SM1 = 01，启动 T1 工作。

进行串行通信发送和接收时采用查询方式，串行发送数据时，单片机不断监测 S2 按键是否按下，一旦 S2 按下则单片机发送字符“BEST”到 PC 端。串行接收数据时，单片机不断监测 RI 标志位是否接收到数据，一旦有数据接收，则将接收到的数据传送到 P0 口，驱动 LED 点亮和熄灭。

为了能够在 PC 端看到单片机发出的数据，借助串口调试助手软件进行观察，读者可以在网络上免费下载计算机串口调试软件，软件运行界面如图 12-6 所示。

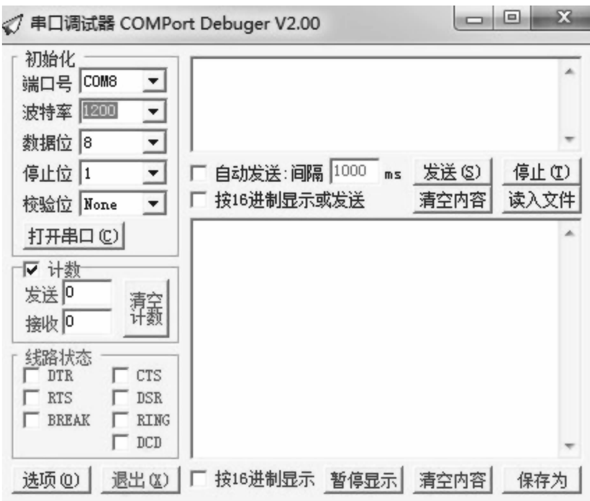



图 12-6　串口调试助手运行界面

通过串口调试助手可以设定串口号、波特率、校验位等参数，但是要注意在实际应用中一定要保证 PC 设置与单片机参数统一，尤其波特率参数设置，否则数据通信将会出错。项目串行通信程序流程图如图 12-7 所示。

在图 12-7 程序流程图的基础上书写串行通信程序代码如下：

```
#include <reg51.h>                      //定义头文件
#include <intrins.h>                    //定义函数头文件
#define uint unsigned int
unsigned char tmp;
char code str[] = {'B','E','S','T',0x00};  //定义发送的字符串
```

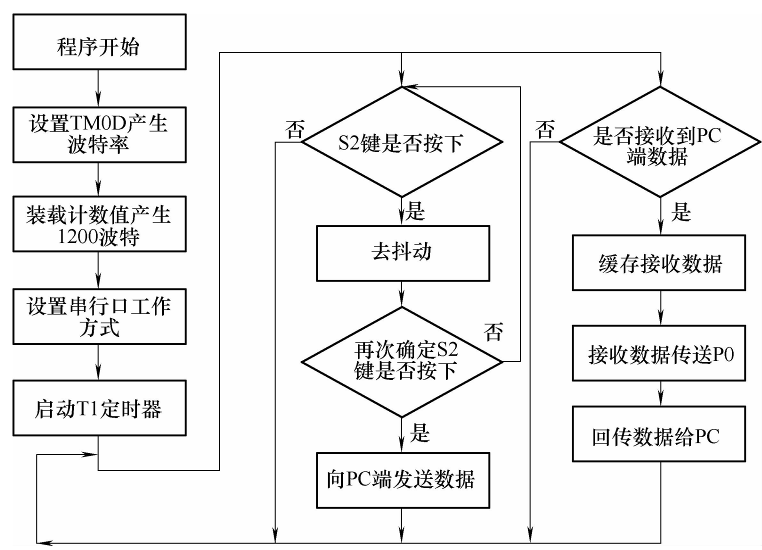



图 12-7　串行通信程序流程图

```
void send_str( ) ;
void delay( unsigned char ms) ;                //定义延迟函数
void send_char( unsigned char txd) ;
sbit S2 = P3^2;                                //定义按键 S2 与单片机 P3. 2 口相连
main( )
{
    TMOD = 0x20;                    //定时器 1 工作于 8 位自动重载模式, 用于产生波特率
    TH1 = 0xE6;                     //波特率 1200
    TL1 = 0xE6;
    SCON = 0x50;                    //设定串行口工作方式
    PCON = 0X00;                    //SMOD = 0
    TR1 = 1;                        //启动定时器 1
    IE = 0x0;                       //禁止任何中断
    while(1)
    {
        if(S2 = =0)                 //扫描按键 S1
        {
            delay(10);              //延时去抖动
            if(S2 = =0)             //再次扫描
            {
                send_str( );        //向 PC 端发送字符串
            }
        }
        if(RI)                      //是否有数据到来
```

```
        {
            RI = 0;
            tmp = SBUF;                  //暂存接收到的数据
            P0 = tmp;                    //数据传送到 P0 口　驱动 LED 点亮和熄灭
            send_char(tmp);              //回传接收到的数据
        }
    }
}
void send_char(unsigned char txd)        //传送一个字符
{
    SBUF = txd;
    while(! TI);                         //等待数据传送
    TI = 0;                              //清除数据传送标志
}
void send_str()                          //传送字串
{
    unsigned char i = 0;
    uint j;
    while(str[i]!= 0x00)
    {
        SBUF = str[i];
        while(TI = = 0);                 //等待数据传送
        TI = 0;                          //清除数据传送标志
        i ++;                            //下一个字符
    }
    for(j = 0;j < 50000;j ++);           //延迟
}
void delay(unsigned char ms)             //延时子程序
{
    unsigned char i;
    while(ms -- )
    {
        for(i = 0;i < 120;i ++);
    }
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 sericom 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 12-8 所示。同时在创建工程路径 sericom 文件夹下生成一个扩展名为 . hex 文件，供下载软件将生成的 . hex 文件下载到单片机中。




图 12-8　程序成功编译结果图

## 12.5　系统调试结果总结

针对串行通信硬件电路和软件程序的设计，在研展科技 YZ200 单片机开发板上调试电路，将 12.4 节中采用查询方式控制定时器 T1 进行串行通信编译生成的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将产生的 .hex 文件烧写到 51 单片机中。通过对开发板通电，用串口线连接好单片机与 PC，打开串行调试助手并配置好参数波特率 1200 波特、数据位 8 位，停止位 1 位，选择可用的串行口如 COM8，在串行调试助手发送区输入十六进制数 F F 0 0 F 0，选择按照十六进制发送，单击发送如图 12-9 所示，则单片机 P0 口驱动 LED 亮灭如图 12-11 所示。按动 S2 按键，则将字符串“BEST”发送到串行调试助手界面的接收区如图 12-10 所示，实现串行通信系统要求。

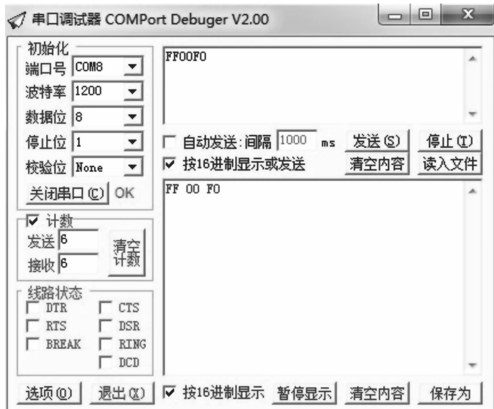



图 12-9　PC 端发送数据

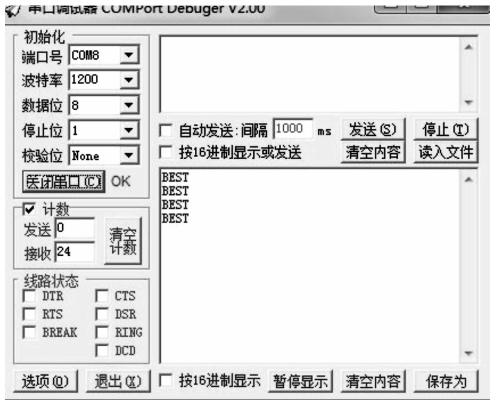



图 12-10　PC 端接收数据

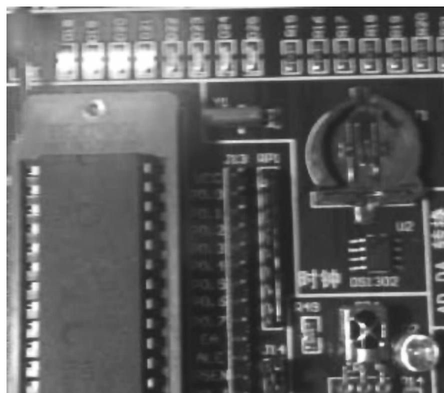



图 12-11　驱动 LED 电路

# 第 13 章　单片机实现 4×4 矩阵键盘控制项目

## 13.1　项目需求

按键对于电子系统来讲属于输入设备，采用按键设计电路时一般有两种方式：一种方式为独立式按键，关于独立按键的使用在第 7 章单片机控制独立按键项目中进行了介绍，另外一种方式为矩阵式按键。例如对于一些电子产品如电子密码锁、手机键盘等一般使用 12～16 个按键进行数据输入，而 4×4 键盘则是一种比较常用的矩阵键盘，矩阵键盘实物如图 13-1 所示。

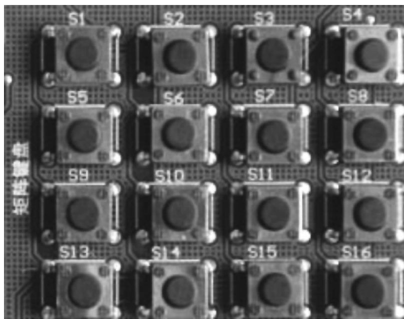



图 13-1　4×4 矩阵键盘实物图

矩阵键盘又称行列键盘，它是用 4 条 I/O 线作为行线，4 条 I/O 线作为列线组成矩阵键盘。在行线和列线的每个交叉点上设置一个按键。这样键盘上按键的个数就为 4×4 个。这种行列式键盘结构能有效地提高单片机系统中 I/O 口的利用率。

为了方便初学者掌握矩阵键盘的使用方法，通过项目实践快速上手。项目要求：设定矩阵键盘编号，按下键盘中的某个按键，可以通过数码管看到显示对应的键盘编号值。

## 13.2　项目工作原理分析

行列式键盘具有广泛的应用，可以采用计算的方法来求出按键值，以得到按键特征码。最常见的键盘键码布局如图 13-2 所示，由 16 个按键组成，单片机中利用 P2 口实现 4×4 矩阵键盘识别功能，单片机与 4×4 矩阵键盘连接的电路结构如图 13-3 所示。

4×4 矩阵键盘的行信号分别接 P2.0～P2.3，列信号分别接 P2.4～P2.7，无按键闭合时，P2.0～P2.3 与 P2.4～P2.7 开路。当有键闭合时，与闭合键相连的两条 I/O 口线之间短路。判断矩阵键盘有按键按下述方法如下：先让 P2.0～P2.3 输出低电平，监测 P2.4～P2.7 的状态，如果 P2.4～P2.7 输出为高电平则没有按键闭合，如果检测到 P2.4～P2.7 输出为低电平则表示有按键按下发生闭合状态，以上只是说明有按键闭合，具体是哪个按键闭合好

需要进一步检测。

一旦确定有按键已经稳定闭合后，接着判断为哪一个按键闭合，采用对按键进行扫描的方式，依次给每一条列线送低电平，其余各列都为高电平，并检测每次扫描的行状态。每当扫描输出某一列为低电平时，相继读入行线状态。若全为高电平，表示为低电平的这列没有按键闭合。否则行线不全部为高电平，表示为低电平的这列上有按键闭合。确定闭合按键的位置后，计算出按键值，产生按键码。

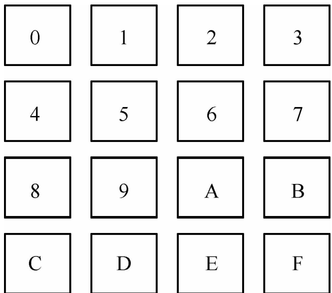



图 13-2　矩阵键盘布局图

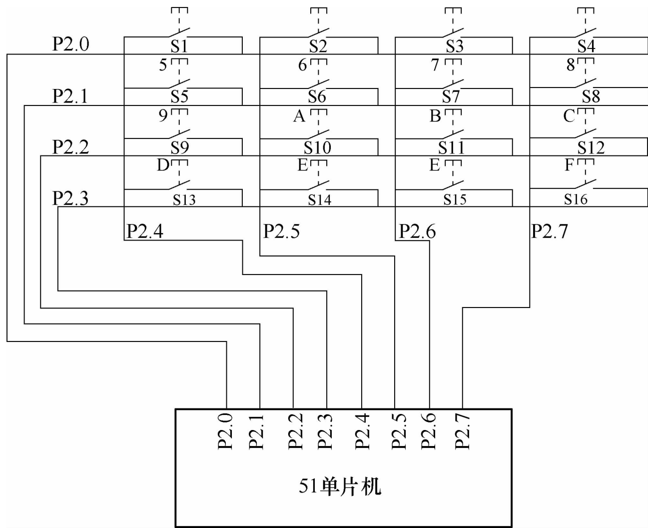



图 13-3　单片机与矩阵键盘接口电路

矩阵按键操作时注意：按键闭合一次只能进行一次按键功能操作，因此须等到按键释放后，再进行键功能操作，否则按一次键，有可能会连续多次进行同样的键操作。

在矩阵按键处理过程中，一旦检测到有按键闭合与确认按键已经稳定闭合期间，通过调用 10 ~20ms 延时子程序避开按键抖动问题。由于按键是机械器件，按下或者松开时有固定的机械抖动，抖动如图 13-4 所示。



图 13-4　按键抖动图

从图 13-4 中可以看出按键按下和松开的瞬间出现机械抖动，这个抖动时间虽然很短，一般 10 ~15ms，不同按键抖动不同，但是对于单片机来讲，由于单片机是 μs 级别，极其容易检测到按键的抖动问题。所以实际上只是进行一次按键操作，但是有可能执行了多次按键结果即为抖动，目前大多数产品实际应用中都使用了按键去抖功能。

按键去抖分为硬件去抖和软件去抖，硬件去抖最简单的是按键两端并联电容，容值根据实验而定。软件去抖使用方便不增加硬件成本，容易调试，所以现在处理按键抖动问题大部分选择软件去抖。软件去抖操作步骤如下：

①检测到按键按下后进行 10 ~ 15ms 延时，用于跳过这个抖动区域。

②延时后再检测按键状态，如果没有按下，表明是抖动或者干扰造成，如果仍旧按下，可以认为是真正的按下，并进行对应的操作。

③同样按键释放后也要进行去抖动延时，延时后检测按键是否真正松开。

多数时候按键需要在释放时才起作用，就类似计算机鼠标一样，这个时候需要检测按键是否释放，如果没有释放则一直等待。基于矩阵按键操作原理，采用单片机控制矩阵按键实现按键键码值显示的电路结构由五部分构成，包括 51 单片机最小系统、数码显示电路、数码驱动电路、矩阵键盘电路和电源供电电路，其电路结构框图如图 13-5 所示。

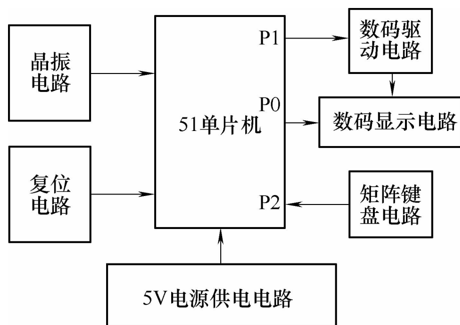



图 13-5　单片机控制矩阵键盘电路结构图

## 13.3　项目硬件电路设计

依据图 13-5 单片机控制矩阵键盘的电路结构框图，进行详细的系统硬件电路设计如图 13-6 所示，电路由单片机主芯片、电源电路、晶振电路、复位电路、数码驱动电路和数码显示电路组成。其中电源电路 5V 供电；晶振电路采用晶体振荡器和微调电容组成，其中电容 C1 和 C2 电容值为 30pF，晶振频率为 11.0592MHz；复位电路采用手动复位。

矩阵键盘电路的行信号分别接 P2.0 ~ P2.3，列信号分别接 P2.4 ~ P2.7，进行按键检测时，假设查询到 P25 为低电平，P24、P26、P27 为高电平，那么可能按下的按键为 S2、S6、S10、S14。进一步探测，先把 P20 设置为低电平，P21、P22、P23 为高电平，如果此时 P25 一直为低电平，就是 S2 按键被按下。如果 P25 为高电平，令其 P21 设为低电平，P20、P22、P23 为高电平，如果此时 P25 为低电平，表明 S6 按键被按下。依此类推，可以确定 S1 ~ S16 中哪个按键被按下。

数码驱动电路采用 8 位三态锁存器 74HC573 芯片用于放大电流驱动数码显示，74HC573 芯片工作时使能信号端 1 引脚 OC 端置低电平，11 引脚 C 端置高电平，输入端 1D ~ 4D 与单片机 P1.0 ~ P1.3 位相连，输出端 1Q ~ 4Q 与集成 4 位数码管位码端 COM1 ~ COM4 相连。集成 4 位数码管段码端 a ~ h 段与单片机 P0.0 ~ P0.7 位相连，选择集成 4 位数码管为共阳型数码管，通过设定数码管段码端（即单片机 P0.0 ~ P0.7 位）为低电平，设定数码管位码端（即单片机 P1.0 ~ P1.3 位）为高电平，就可以在数码管上显示对应的按键数码。

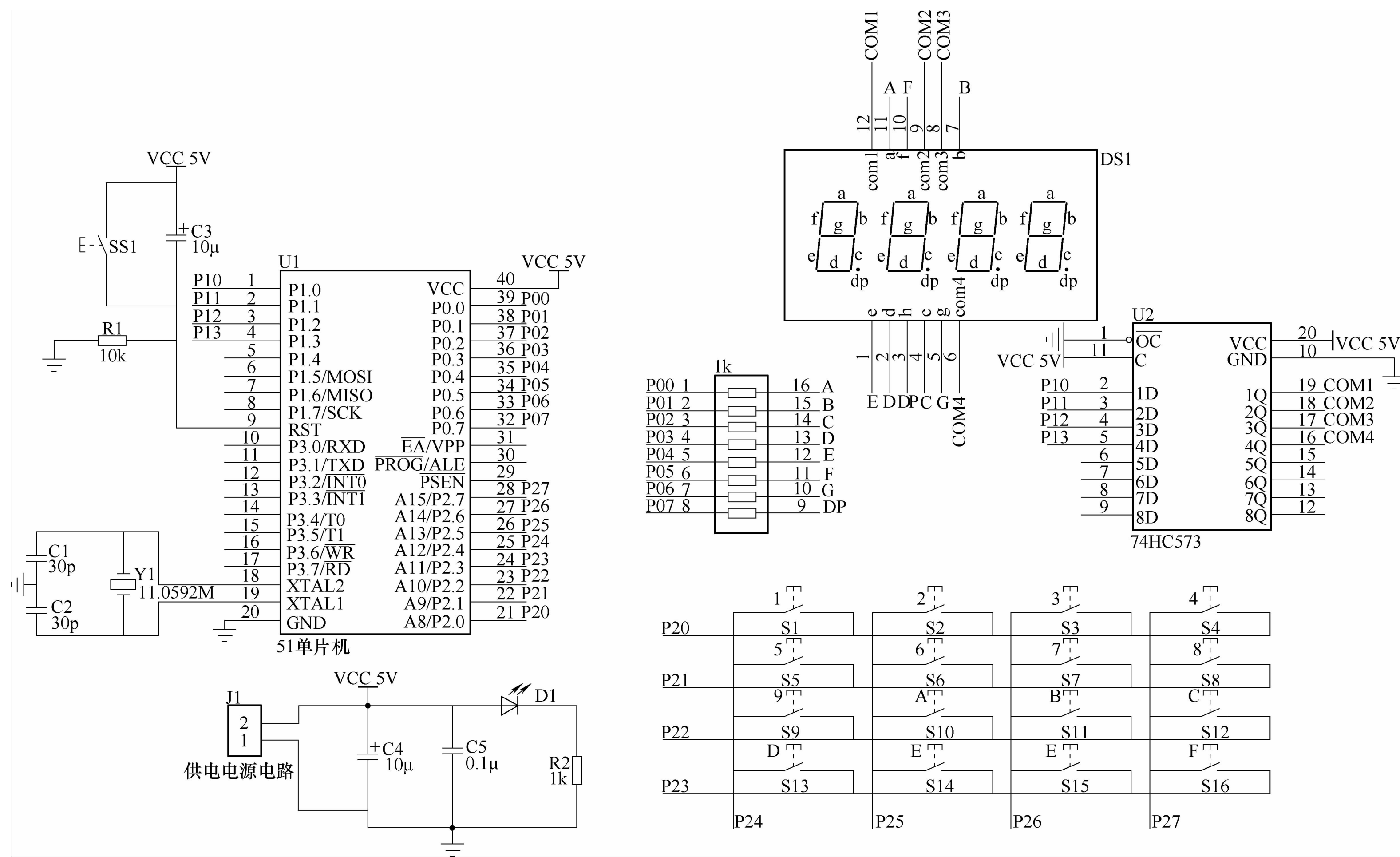



图13-6　单片机控制矩阵键盘硬件电路原理图

## 13.4　项目软件程序设计

项目通过按下相应键后在 4 位集成数码管上显示出对应按键值。0 ~ 16 个按键值在数码管分别对应显示 0 ~ F。根据图 13.6 矩阵键盘电路原理图，矩阵键盘行线 P20 ~ P23 为输出线，列线 P24 ~ P27 为输入线。单片机将行线（P20 ~ P23）全部输出低电平，此时读入列线数据，若列线全为高电平则没有键按下，当列线有出现低电平时调用延时程序以此来去除按键抖动。延时完成后再判断是否有低电平，如果此时读入列线数据还是有低电平，则说明确实有键按下，再来进一步确定键值。如以第二行的 S7 按键为例，若按下 S7 按键后，先判断是否有按键按下，一旦判断确实有按键按下后，行线轮流输出低电平，根据读入列线的数据可以确定键值。首先，单片机将 P20 输出为低电平，其他 P21 ~ P23 输出高电平，此时读取列线的数据全为高电平，说明没有在第一行有键按下；其次，单片机将 P21 输出低电平，其他 P20、P22、P23 仍为高电平，此时再来读取列线数据，发现列线读到的数据有低电平，数值为 P20 = 1、P21 = 0、P22 = 1、P23 = 1（即 1101），此时列项 S7 按键按下导致 P27 = 0、P26 = 1、P25 = 1、P24 = 1（即 0111），最后将按键特征码组合为 01111101（即 0x7d），那么 0x7d 代表 S7 的按键值，执行按键扫描功能处理子程序就可以达到目的。依据上述原理矩阵键盘控制系统软件设计流程图如图 13-7 所示。

单片机控制矩阵按键，通过按不同的按键在数码管端显示对应的按键码程序如下：

```
#include <reg52.h>    //包含头文件,头文件包含特殊功能寄存器的定义
#define uchar unsigned char
#define uint   unsigned int
unsigned char const table[ ] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
          0x80,0x90,0x88,0x83,0xa7,0xa1,0x86,0x8e}; //定义共阳数码管段码表 0 ~ F
uchar keyscan(void); //键盘扫描函数声明
void delay(void);      //延时函数声明
void main( )           //定义主函数
  {
    uchar key;
    P1 = 0xff;           //设置数码管位码端电平,显示按键上的按键码
    while(1)
  {
    key = keyscan( );  //调用键盘扫描,
    switch(key)
    {
      case 0xee:P0 = table[0];break;   //显示按键码“0”
      case 0xde:P0 = table[1];break;   //显示按键码“1”
      case 0xbe:P0 = table[2];break;   //显示按键码“2”
      case 0x7e:P0 = table[3];break;   //显示按键码“3”
      case 0xed:P0 = table[4];break;   //显示按键码“4”
```

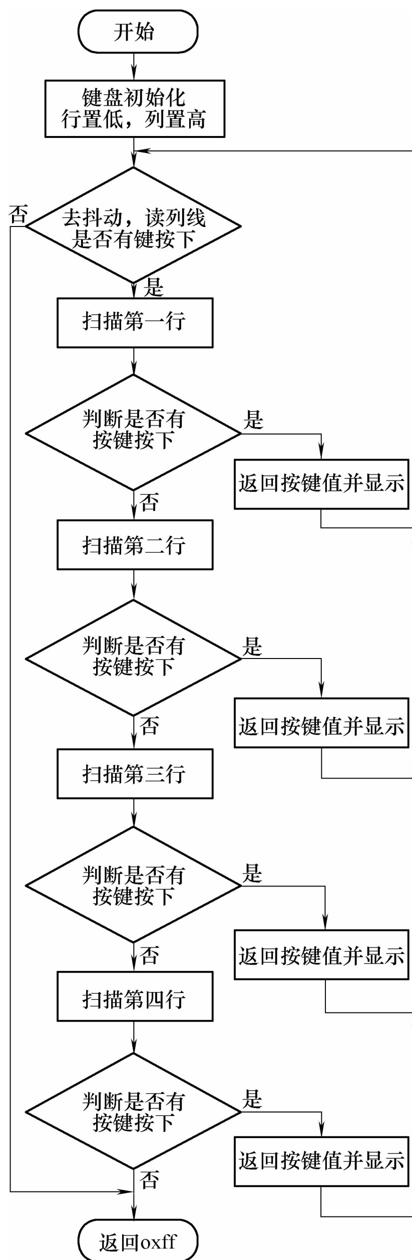



图 13-7　系统软件设计流程图

```
case 0xdd:P0 = table[5];break;   //显示按键码"5"
case 0xbd:P0 = table[6];break;   //显示按键码"6"
case 0x7d:P0 = table[7];break;   //显示按键码"7"
case 0xeb:P0 = table[8];break;   //显示按键码"8"
case 0xdb:P0 = table[9];break;   //显示按键码"9"
```

```
        case 0xbb:P0 = table[10];break; //显示按键码“a”
        case 0x7b:P0 = table[11];break; //显示按键码“b”
        case 0xe7:P0 = table[12];break; //显示按键码“c”
        case 0xd7:P0 = table[13];break; //显示按键码“d”
        case 0xb7:P0 = table[14];break; //显示按键码“e”
        case 0x77:P0 = table[15];break; //显示按键码“f”
      }
    }
}
  uchar keyscan(void)          //键盘扫描函数
  {
    uchar key_h,key_l;         //行列值中间变量
    P2 =0xf0;                  //行线输出全为 0
    key_h = P2&0xf0;           //读入列线值
    if(key_h!  =0xf0)          //先检测有无按键按下
    {
  delay();            //去抖
  if(key_h!  =0xf0)
  {
    key_h = P2&0xf0;   //读入列线值
    P2 =key_h|0x0f;    //输出当前列线值
    key_l = P2&0x0f;   //读入行线值
    return(key_h + key_l);//键盘最后组合码值
    }
  }
return(0xff);        //返回该值
}
void delay(void)       //延时函数
{
    unsigned char i,j;
    for(i =0;i <20;i ++)
    for(j =0;j <250;j ++);
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 juzhenkey 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 13-8 所示。同时在创建工程路径 juzhenkey 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。

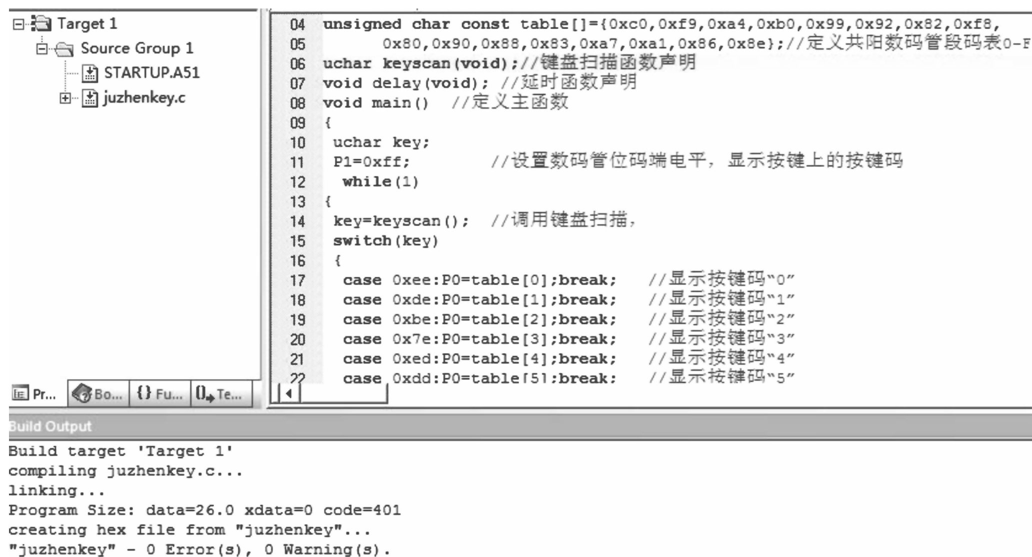



图 13-8　程序成功编译结果图

## 13.5　系统调试结果总结

采用单片机控制矩阵键盘实现对应按键值在数码管上显示，其系统的硬件电路和软件程序在研展科技 YZ200 单片机开发板上进行调试，将 13.4 节 Keil 编译完成后产生的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将 .hex 文件烧写到 51 单片机中。通过给开发板上电，按动矩阵键盘的 S13 按键（对应的按键码为“d”），则对应的 4 位集成数码管直接显示对应的按键值“d”，实现项目要求。

# 第 14 章　单片机实现字符型液晶显示项目

## 14.1　项目需求

小型智能化电子产品设计时，普通的 7 段 LED 数码管只能用来显示数字，如果需要显示英文字符、图像或者汉字时，则必须选择使用液晶显示器（简称 LCD）。在实际应用中计算器、万用表、电子表及很多家用电子产品中都采用液晶显示数字、符号和图形等。LCD 可分为两种类型，一种是字符模式 LCD，另一种是图形模式 LCD，本章主要介绍 16×2 字符型点阵式 LCD。市场上有各种不同品牌的字符显示类型的 LCD，本章选用字符型 LCD1602 为例，介绍其用法。一般 1602 字符型液晶实物如图 14-1 所示。



图 14-1　LCD1602 字符型液晶实物图

通过单片机控制液晶显示项目操作，使得读者可以快速掌握字符型液晶的使用方法，设计要求为单片机 P0.0～P0.7 与液晶的数据 I/O 口相连，1602 液晶显示内容如下：

good good study

day day up, 88.

## 14.2　项目工作原理

液晶显示的原理是利用液晶的物理特性，通过电压对其显示区域进行控制，有电就有显示，这样即可以显示出图形。液晶显示器具有厚度薄、适用于大规模集成电路直接驱动、目前已经被广泛地应用在便携式计算机、数字摄像机、PDA 移动通信工具等众多领域。

字符型液晶显示模块是一种专门用于显示字母、数字、符号等点阵式 LCD 显示模块，分 4 位和 8 位数据传输方式，提供 5×7 点阵 + 光标和 5×10 点阵 + 光标的显示模式。提供显示数据缓冲区 DDRAM、字符发生器 CGROM 和字符发生器 CGRAM，可以使用 CGRAM 来存储自己定义的最多 8 个 5×8 点阵的图形字符的字模数据。提供丰富的指令设置：清显示、光标回原点、显示开/关、光标开/关、显示字符闪烁、光标移位、显示移位等。提供内部上电自动复位电路，当外加电源电压超过 4.5V 时，自动对模块进行初始化操作，将模块设置为默认的显示工作状态。

16×2 点阵字符液晶模块是由点阵字符液晶显示器件和专用的行、列驱动器，控制器以及必要的连接件，结构件装配而成，可以显示数字和英文字符，这种点阵字符模块本身带有字符以及发生器，显示容量大，功能丰富。

1602LCD 分为带背光和不带背光两种，其控制器大部分为 HD44780，带背光的比不带背光的厚，是否带背光在应用中并无差别，两者尺寸差别如图 14-2 所示。1602LCD 芯片逻

辑工作电压 4.5 ~5.5V，LCD 驱动电压 3 ~5V，最佳工作电压 5.0V，显示容量 16 ×2 个字符；工作电流 2.0mA；字符尺寸 2.95 ×4.35(W×H)mm。

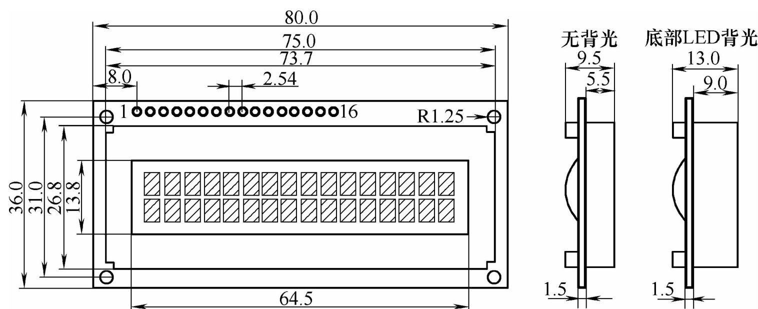



图 14-2　1602LCD 尺寸图

从图 14-2 中可以看到 1602LCD 采用标准的 14 引脚（无背光）或 16 引脚（带背光）接口，各引脚描述见表 14-1。

**表 14-1　LCD1602 引脚描述**

| 符　号 | 引 脚 说 明 | 符　号 | 引 脚 说 明 |
|---|---|---|---|
| VSS | 电源地 | D2 | 数据 |
| VDD | 电源正极 | D3 | 数据 |
| VL | 液晶显示偏压 | D4 | 数据 |
| RS | 数据/命令选择 | D5 | 数据 |
| R/W | 读/写选择 | D6 | 数据 |
| E | 使能信号 | D7 | 数据 |
| D0 | 数据 | BLA | 背光源正极 |
| D1 | 数据 | BLK | 背光源负极 |

第 1 引脚：VSS 为地电源。

第 2 引脚：VDD 接 5V 正电源。

第 3 引脚：VL 为液晶显示器对比度调整端，接正电源时对比度最弱，接地时对比度最高，对比度过高时会产生“鬼影”，使用时可以通过一个 10kΩ 的电位器调整对比度。

第 4 引脚：RS 为寄存器选择，RS =0 当单片机进行读模块操作，指向地址计数器。当单片机进行写模块操作，指向指令寄存器。当 RS =1 时，无论单片机读/写操作，均指向数据寄存器。

第 5 引脚：R/W 为读写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 R/W 共同为低电平时可以写入指令或者显示地址，当 RS 为低电平，R/W 为高电平时，可以读忙信号，当 RS 为高电平，R/W 为低电平时，可以写入数据。

第 6 引脚：E 端读操作时，信号下降沿有效，写操作时，高电平有效。

第 7 ~14 引脚：单片机与液晶之间的数据传送通道。

第 15 引脚：背光电源正极（5V）。

第 16 引脚：背光电源负极（0V）。

LCD1602 基本操作时序见表 14-2。

**表 14-2 LCD1602 基本操作时序表**

| | | | | |
|---|---|---|---|---|
| 读状态 | 输入 | RS = L, R/W = H, E = H | 输出 | D0 ~ D7 = 状态字 |
| 写指令 | 输入 | RS = L, R/W = L, D0 ~ D7 = 指令码, E = 高脉冲 | 输出 | 无 |
| 读数据 | 输入 | RS = H, R/W = H, E = H | 输出 | D0 ~ D7 = 数据 |
| 写数据 | 输入 | RS = H, R/W = L, D0 ~ D7 = 数据, E = 高脉冲 | 输出 | 无 |

1602 液晶模块内部的字符发生存储器（CGROM）已经存储了 192 个 5 ×7 的点矩阵字型，CGROM 的字型经过内部电路的转换传到显示器上，仅能读出不可以写入。字型或者字符的排列方式与标准的 ASCII 码相同，例如字符码 31H 为“1”字符，字符码 41H 为“A”字符。将 A 的 ASCII 码 01000001B（41H）写入到 DDRAM 中，同时电路到 CGROM 中将 A 的字型点阵数据找出来显示在 LCD 上，就能看到字母“A”，字符与字符码对照表见表 14-3。

**表 14-3 字符与字符码对应图**

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| xxxx0000 | CG RAM (1) | | | 0 | @ | P | ` | p | | | | ー | タ | ミ | α | p |
| xxxx0001 | (2) | | ! | 1 | A | Q | a | q | | | 。 | ア | チ | ム | ä | q |
| xxxx0010 | (3) | | " | 2 | B | R | b | r | | | 「 | イ | ツ | メ | β | θ |
| xxxx0011 | (4) | | # | 3 | C | S | c | s | | | 」 | ウ | テ | モ | ε | ∞ |
| xxxx0100 | (5) | | $ | 4 | D | T | d | t | | | 、 | エ | ト | ヤ | μ | Ω |
| xxxx0101 | (6) | | % | 5 | E | U | e | u | | | ・ | オ | ナ | ユ | σ | ü |
| xxxx0110 | (7) | | & | 6 | F | V | f | v | | | ヲ | カ | ニ | ヨ | ρ | Σ |
| xxxx0111 | (8) | | ' | 7 | G | W | g | w | | | ァ | キ | ヌ | ラ | g | π |
| xxxx1000 | (1) | | ( | 8 | H | X | h | x | | | ィ | ク | ネ | リ | √ | x̄ |
| xxxx1001 | (2) | | ) | 9 | I | Y | i | y | | | ゥ | ケ | ノ | ル | ⁻¹ | y |
| xxxx1010 | (3) | | * | : | J | Z | j | z | | | ェ | コ | ハ | レ | j | 千 |
| xxxx1011 | (4) | | + | ; | K | [ | k | { | | | ォ | サ | ヒ | ロ | ˣ | 万 |
| xxxx1100 | (5) | | , | < | L | ¥ | l | \| | | | ャ | シ | フ | ワ | ¢ | 円 |
| xxxx1101 | (6) | | - | = | M | ] | m | } | | | ュ | ス | ヘ | ン | £ | ÷ |
| xxxx1110 | (7) | | . | > | N | ^ | n | → | | | ョ | セ | ホ | ゛ | ñ | |
| xxxx1111 | (8) | | / | ? | O | _ | o | ← | | | ッ | ソ | マ | ゜ | ö | █ |

由于单片机可以直接访问模块内部的 IR 和 DR，作为缓冲区域，IR 和 DR 在模块进行内部操作之前，可以暂存来自 MPU 的控制信息，这样就给用户在单片机和外围控制设备的选择上，增加了余地。模块的内部操作由单片机的 RS、R/W、E 以及数据信号 DB0 ~ DB7 决定，这些信号的组合形成了显示模块的指令。LCD 模块向用户提供 11 条指令，大致可以分为

（1）清楚显示器；

（2）光标归位设定；

（3）设定字符进入模式；

（4）显示器开关；

（5）显示光标移位；

（6）功能设定；

（7）CGRAM；

（8）DDRAM 地址设定；

（9）忙碌标志 BF 或 AC 地址读取；

（10）写数据到 CGRAM 或者 DDRAM 中；

（11）从 CGRAM 或者 DDRAM 中读取数据。

一般情况下，内部 RAM 数据传送的动能使用最为频繁，因此，RAM 中的地址指针具备自动加一或者减一功能，在一定程度上减轻了单片机编程负担。此外，由于数据移位指令与写显示数据可同时进行，用户能够以最少系统开发时间，达到最高的编程效率。编写代码时，单片机每次访问 LCD 显示模块之前，单片机应首先监测忙碌标志 BF，确认 BF = 0，访问过程才能进行。具体指令操作可以查询 LCD1602 字符型液晶显示模块使用说明书。

## 14.3　项目硬件电路设计

1602 液晶显示模块可以和 51 单片机直接相连，电路如图 14-3 所示。单片机的 P0.0 ~ P0.7 与液晶数据输入、输出口 DB0 ~ DB7 相连，实现 8 位双向数据传输，它的方向由读写控制引脚 R/W 来决定，高电平为读，此时数据出现在总线上，可以由 CPU 读走；低电平为写，可以写入 8 位数据。E 为使能信号引脚，高电平有效。图 14-3 中 10kΩ 滑动变阻器用来调节提供给驱动器的供电电压，从而调节液晶显示的对比度。

图 14-3　液晶显示硬件电路原理图

## 14.4　项目软件设计

程序的功能是在液晶屏上实现“good good study”“day day up，88”字符显示。程序设计的重点在于对液晶显示模块的驱动，由于 1602 内部含有控制驱动器，需要了解控制器的指令集并掌握如何使用。关于 1602 指令集可以查询 1602 液晶使用说明手册均给出详细的描述过程。液晶显示程序流程图如图 14-4 所示。

在图 14-4 基础上书写 1602 液晶显示软件程序代码如下：

```
#include <reg51.h>   //包含头文件
#include <intrins.h>
typedef unsigned char BYTE;   //数据类型的宏定义
typedef unsigned int WORD;
typedef bit BOOL ;
sbit rs = P2^0;       //定义液晶的使能信号 rs R/W E 分别与单片机的 P2.0 ~ P2.2 相连
sbit rw = P2^1;
sbit ep = P2^2;
BYTE code dis1[] = {"good good study"};
BYTE code dis2[] = {"day day up,88"};
```

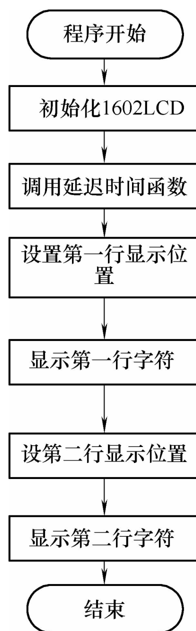



图 14-4 1602 液晶显示程序设计流程图

```
void delay(BYTE ms)
{                                       //延时子程序
    BYTE i;
    while(ms --)
        {
        for(i = 0; i < 250;i ++)
        {
            _nop_();
            _nop_();
            _nop_();
            _nop_();
        }
    }
}
BOOL lcd_bz()
{                                       //测试 LCD 忙碌状态
    BOOL result;
    rs = 0;
    rw = 1;
    ep = 1;
    _nop_();
```

```
    _nop_();
    _nop_();
    _nop_();
    result = (BOOL)(P0 & 0x80);
    ep = 0;
    return result;
}
void lcd_wcmd(BYTE cmd)
{                                           //写入指令数据到 LCD
    while(lcd_bz());
    rs = 0;
    rw = 0;
    ep = 0;
    _nop_();
    _nop_();
    P0 = cmd;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    ep = 1;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    ep = 0;
}
void lcd_pos(BYTE pos)
{                                           //设定显示位置
    lcd_wcmd(pos | 0x80);
}
void lcd_wdat(BYTE dat)
{                                           //写入字符显示数据到 LCD
    while(lcd_bz());
    rs = 1;
    rw = 0;
    ep = 0;
    P0 = dat;
    _nop_();
```

```
    _nop_();
    _nop_();
    _nop_();
    ep = 1;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    ep = 0;
}
void lcd_init()
{                                    //LCD 初始化设定
    lcd_wcmd(0x38);
    delay(1);
    lcd_wcmd(0x0c);
    delay(1);
    lcd_wcmd(0x06);
    delay(1);
    lcd_wcmd(0x01);                  //清除 LCD 的显示内容
    delay(1);
}
main()
{
    BYTE i;
    lcd_init();                      //初始化 LCD
    delay(10);
    lcd_pos(4);                      //设置显示位置为第一行的第 5 个字符
    i = 0;
    while(dis1[i]! = '\0')
    {                                //显示字符
        lcd_wdat(dis1[i]);
        i++;
    }
    lcd_pos(0x41);                   //设置显示位置为第二行第 2 个字符
    i = 0;
    while(dis2[i]!='\0')
    {
        lcd_wdat(dis2[i]);           //显示字符
        i++;
```

```
    }
    while(1);
}
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 yejing 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 14-5 所示。同时在创建工程路径 yejing 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。




图 14-5　液晶程序编译成功界面

## 14.5　项目调试

针对 1602 液晶显示硬件电路和软件程序，在研展科技 YZ200 单片机开发板上调试 1602 液晶显示电路，将 14.4 节 Keil 编译完成后产生的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将 .hex 文件烧写到 51 单片机中。开发板通电，可以看到液晶显示屏幕显示“good good study”“ day day up，88”，显示成功。作为一种输出方式，液晶显示最大的优点是能够实现友好的人机界面。液晶显示广泛应用于工业控制和智能化仪器仪表等领域，成为单片机应用开发领域的常用典型模块之一。

# 第 15 章　单片机实现步进电动机控制项目

## 15.1　项目需求

在工业控制系统中，通常要控制机械部件的平移和转动，这些部件的驱动大都采用交流电动机，直流电动机和步进电动机。在这三种电动机中，步进电动机最适合于数字控制，步进电动机广泛应用于对精度要求比较高的运动控制系统中，如机器人、机械阀门控制器等。一般电动机都是连续旋转，而步进电动机是一步一步转动，每当步进电动机驱动器收到一个驱动脉冲信号，步进电动机将会按照设定的方向转动一个固定的角度，因此步进电动机是一种将电脉冲转化为角位移的执行机械。对于角位移步进电动机，用户可以通过控制脉冲的个数来控制角位移量，从而达到准确定位的目的；另外也可以通过控制脉冲频率控制电动机转动的速度和加速度，达到调速的目的。本章重点介绍步进电动机的原理及应用，通过 51 单片机控制四相步进电动机转动系统，主要由单片机产生正确的驱动脉冲信号，控制步进电动机通过正确的转速正转或者反转产生正确的转动角度。

## 15.2　项目工作原理分析

步进电动机是将电脉冲信号转变为角位移或线位移的开环控制元件。在非超载的情况下，电动机的转速、停止的位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，当步进驱动器接收到一个脉冲信号，它就驱动步进电动机按设定的方向转动一个固定的角度（称为“步距角”），它的旋转是以固定的角度一步一步运行的。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时可以通过控制脉冲频率来控制电动机转动的速度和加速度，从而达到调速的目的。

现在比较常用的步进电动机分为三种：反应式步进电动机（VR）、永磁式步进电动机（PM）、混合式步进电动机（HB）。本章节以反应式步进电动机为例，介绍其基本原理与应用方法。反应式步进电动机可实现大转矩输出，步进角一般为 1.5°。反应式步进电动机的转子磁路由软磁材料制成，定子上有多相励磁绕组，利用磁导的变化产生转矩。常用小型步进电动机实物如图 15-1 所示。

在应用选型时可以根据电动机的不同参数来决定应用范围，一般步进电动机主要由以下几个参数决定：转矩、步矩角、相数、保持转矩、静转矩、拍数。



图 15-1　步进电动机实物图

**1. 转矩**

电动机一旦通电，在定转子间将产生磁场（磁通量 $\Phi$）当转子与定子错开一定角度 $\theta$ 产生力 $F$ 与（$\mathrm{d}\Phi/\mathrm{d}\theta$）成正比。磁通量为

$$\Phi = B_r S$$

式中，$B_r$ 为磁通密度；$S$ 为导磁面积。

$F$ 与 $LDB_r$ 成正比，其中 $L$ 为铁心有效长度，$D$ 为转子直径。

$$B_r = NI/R$$

式中，$NI$ 为励磁绕组安匝数（电流乘匝数）；$R$ 为磁阻。

$$转矩 = 力 \times 半径$$

转矩与电动机有效体积、安匝数、磁通密度成正比。因此，电动机有效体积越大，励磁安匝数越大，定转子间气隙越小，电动机转矩越大，反之亦然。

**2. 步距角**

电动机固有步距角表示控制系统每发一个步进脉冲信号，电动机所转动的角度。一般电动机出厂时都给出了一个步距角的值，如 86BYG250A 型电动机给出的值为 0.9°/1.8°（表示半步工作时为0.9°、整步工作时为1.8°），这个步距角可以称之为‘电动机固有步距角’，它不一定是电动机实际工作时的真正步距角，真正的步距角还和驱动器有关。电动机转子转过的角位移用 $\theta$ 表示。$\theta = 360°/$（转子齿数 J × 运行拍数），以常规两、四相，转子齿为 50 齿电动机为例。四拍运行时步距角为 $\theta = 360°/(50 \times 4) = 1.8°$（俗称整步），八拍运行时步距角为 $\theta = 360°/(50 \times 8) = 0.9°$（俗称半步）。

**3. 相数**

步进电动机的相数是指电动机内部产生不同对极 N、S 磁场的励磁绕组对数，常用 $m$ 表示。目前常用的有两相、三相、四相、五相步进电动机。电动机相数不同，其步距角也不同，一般两相电动机的步距角为 0.9°/1.8°、三相的为 0.75°/1.5°、五相的为 0.36°/0.72°。在没有细分驱动器时，用户主要靠选择不同相数的步进电动机来满足自己步距角的要求。

**4. 保持转矩**

保持转矩是指步进电动机通电但没有转动时，定子锁住转子的转矩。它是步进电动机最重要的参数之一，通常步进电动机在低速时的转矩接近保持转矩。由于步进电动机的输出转矩随速度的增大而不断衰减，输出功率也随速度的增大而变化，所以保持转矩就成为了衡量步进电动机最重要的参数之一。如当人们说 2N · m 的步进电动机，在没有特殊说明的情况下是指保持转矩为 2N · m 的步进电动机。

**5. 静转矩**

静转矩表示电动机在额定静态电作用下，电动机不作旋转运动时，电动机转轴的锁定转矩。此转矩是衡量电动机体积（几何尺寸）的标准，与驱动电压及驱动电源等无关。

**6. 拍数**

步进电动机拍数是指完成一个磁场周期性变化所需脉冲数或导电状态，用 $n$ 表示，或指电动机转过一个齿距角所需脉冲数，以四相电动机为例，有四相四拍运行方式即 AB-BC-CD-DA-AB，四相八拍运行方式即 A-AB-B-BC-C-CD-D-DA-A。

步进电动机的励磁方式一般分为 1 相励磁、2 相励磁、1-2 相励磁。其中 1 相励磁最为简单，转矩最小；2 相励磁可有较大的转矩；1-2 相励磁是属于半步的方式，也就是说旋转角度为前两种方式的一半。下面对三种励磁方式归类如下，1 相励磁见表 15-1，2 相励磁见表 15-2，1-2 相励磁见表 15-3。

1 相励磁法：每一瞬间只有一个绕组导通，其他绕组休息。其特点是励磁方法简单，耗电低，精确度良好。但是转矩小，振动大，每次励磁信号走的角度是标称角度。

2 相励磁法：每一瞬间有两个绕组同时导通，特点是转矩大，振动小，每次励磁转动角度是标称角度。

表 15-1　1 相励磁

| 步数 | A | B | /A | /B | 步数 | A | B | /A | /B |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 5 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 7 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 1 |

表 15-2　2 相励磁

| 步数 | A | B | /A | /B | 步数 | A | B | /A | /B |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 5 | 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 7 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 8 | 1 | 0 | 0 | 1 |

表 15-3　1-2 相励磁

| 步数 | A | B | /A | /B | 步数 | A | B | /A | /B |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 5 | 0 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 0 | 6 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 7 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 | 8 | 1 | 0 | 0 | 1 |

1-2 相励磁法：1 相和 2 相轮流交替导通，精度较高，且运作平滑。每送一个励磁信号转动二分之一标称角度。又称为半步驱动。

步进电动机的驱动电路依据控制信号工作，控制信号由单片机产生，完成以下三种功能：

①控制换相顺序，通电换相称为脉冲分配，对于四相步进电动机而言，其各相通电顺序按照 A-B-$\overline{A}$-$\overline{B}$，通电控制脉冲必须严格按照顺序执行。

②控制步进电动机的转向，如果按照给定工作方式的正序通电换相，步进电动机正转；如果按照反序通电换相，电动机反转。

③控制步进电动机的速度，如果给步进电动机发一个控制脉冲，它就转一步，再发一个脉冲，它会再转一步。两个脉冲间隔越短，步进电动机转得越快。调整单片机发出脉冲频率，即可以对步进电动机进行调速。

## 15.3　项目硬件电路设计

完整的步进电动机控制系统包括控制器、驱动器、电动机三部分构成。系统结构框如图 15-2 所示。

步进电动机正常工作需要提供具有一定驱动能力的脉冲信号，脉冲信号由单片机输出的激励信号经过脉冲分配器




图 15-2　步进电动机控制系统框图

产生，脉冲分配可以通过软件方便灵活地实现，也可以由硬件脉冲分配电路实现。随着大规模集成电路技术的发展，现在很多厂家生产硬件脉冲分配与驱动器集成在一起的芯片，方便电路设计。本设计涉及小型步进电动机驱动电路选用 ULN2003，ULN2003 是高压大电流达林顿晶体管阵列系列产品，具有电流增益高、工作电压高、温度范围宽、带负载能力强等特点，适应于各类要求高速大功率驱动的系统，ULN2003 内部结构及等效电路图如图 15-3 所示。

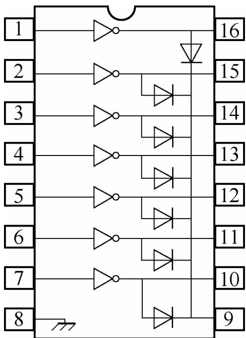


图 15-3　ULN2003 引脚图

引脚介绍如下：

引脚 1：CPU 脉冲输入端，端口对应一个信号输出端。

引脚 2：CPU 脉冲输入端。

引脚 3：CPU 脉冲输入端。

引脚 4：CPU 脉冲输入端。

引脚 5：CPU 脉冲输入端。

引脚 6：CPU 脉冲输入端。

引脚 7：CPU 脉冲输入端。

引脚 8：接地。

引脚 9：该脚是内部 7 个续流二极管负极的公共端，各二极管的正极分别接各达林顿管的集电极。用于感性负载时，该脚接负载电源正极，实现续流作用。如果该脚接地，实际上就是达林顿管的集电极对地接通。

引脚 10：脉冲信号输出端，对应 7 脚信号输入端。

引脚 11：脉冲信号输出端，对应 6 脚信号输入端。

引脚 12：脉冲信号输出端，对应 5 脚信号输入端。

引脚 13：脉冲信号输出端，对应 4 脚信号输入端。

引脚 14：脉冲信号输出端，对应 3 脚信号输入端。

引脚 15：脉冲信号输出端，对应 2 脚信号输入端。

引脚 16：脉冲信号输出端，对应 1 脚信号输入端。

单片机控制电动机正、反转电路采用 ULN2003 芯片驱动步进电动机，驱动电流放大，其中单片机的 P2.0 ~ P2.3 口与 ULN2003 相连，ULN2003 输出引脚 16、15、14、13 连接到步进电动机端，驱动电动机转动。

电动机正、反转的环形脉冲分配表见表 15-4、表 15-5。

**表 15-4　电动机正转环形脉冲分配表**

| 步　　数 | P20 | P21 | P22 | P23 | 步　　数 | P20 | P21 | P22 | P23 |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | $\overline{A}$ | $\overline{B}$ | | A | B | $\overline{A}$ | $\overline{B}$ |
| 1 | 1 | 1 | 0 | 0 | 3 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 | 0 | 4 | 1 | 0 | 0 | 1 |

**表 15-5　电动机反转环形脉冲分配表**

| 步数 | P20 | P21 | P22 | P23 | 步数 | P20 | P21 | P22 | P23 |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | $\overline{A}$ | $\overline{B}$ | | A | B | $\overline{A}$ | $\overline{B}$ |
| 1 | 1 | 1 | 0 | 0 | 3 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 | 4 | 0 | 1 | 1 | 0 |

其设计的电动机正、反转硬件电路如图 15-4 所示。

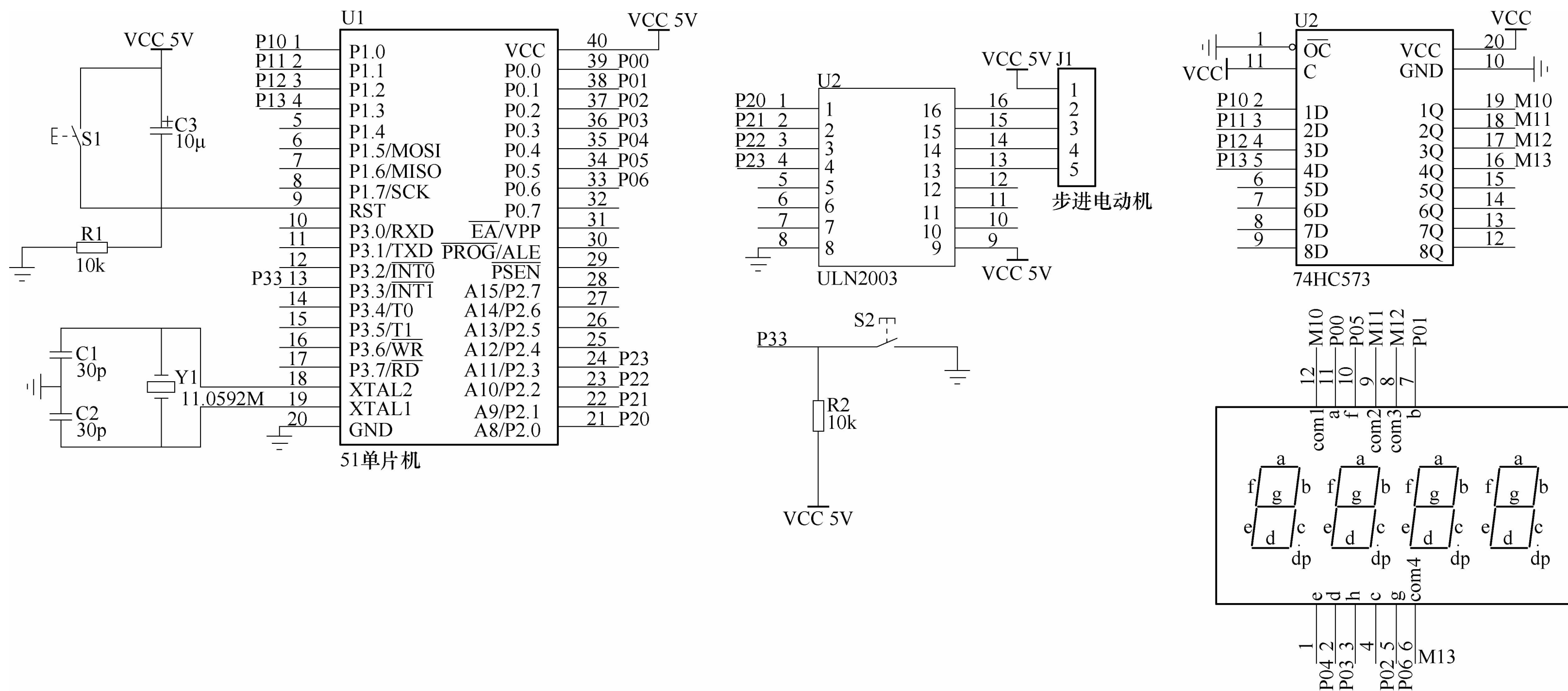



图15-4　电动机正、反转硬件电路原理图

## 15.4　项目软件设计

单片机通过对按键的操作实现步进电动机正、反转和停止操作，采用定时器 T1 利用中断的方式实现对每次按键触发动作进行监控，一旦检测到有按键按下，按键标志位 sign 会进行计数，形成按键标志 sign =0、sign =1、sign =2，单片机通过对按键标志位 sign 计数判断，实现电动机转动。例如：如果判断 sign =0 实现电动机正转，如果判断 sign =1 实现电动机反转，如果判断 sign =2 则控制电动机停止转动。

对于步进电动机的转动考虑到步进电动机系统中有脉冲分配电路和驱动电路两个重要电路，脉冲分配电路有步进脉冲和转向控制两个输入信号，脉冲分配电路在步进脉冲信号和转向控制信号的共同作用下产生正确转向的四相激励信号，此激励信号经过驱动电路送至步进电机，从而控制步进电动机按照指定方向转动，此激励信号的频率决定了步进电动机的转速。其单片机控制步进电动机转向的程序流程如图 15-5 所示。

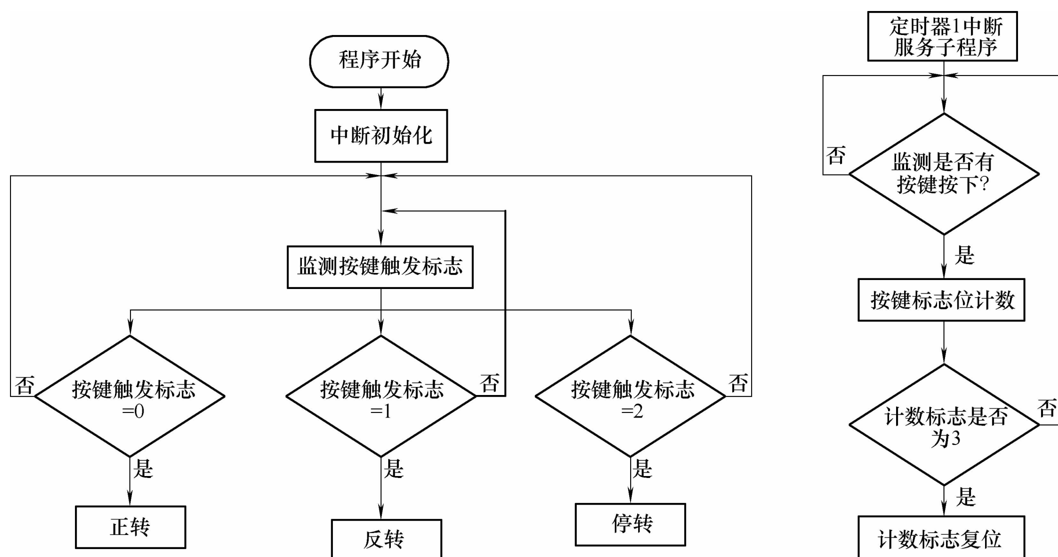



图 15-5　单片机控制步进电动机程序流程图

依据图 15-5 程序设计思想单片机控制步进电动机实现正、反转和停止，程序代码如下：

```
#include  <reg52. h >
unsigned char sign;       //定义正、反转和停止标志位
sbit key = P3^3;          //定义按键接口
unsigned char code ctable[4] = {0xf1,0xf2,0xf4,0xf8};      //正转表
unsigned char code etable[4] = {0xf8,0xf4,0xf2,0xf1};      //反转表
void delay(unsigned int i)   //延时
{
while( --i);
}
```

```
void main()
{
    unsigned char i;
    EX1 =1;          //外部中断0 开
    IT1 =1;          //边沿触发
    EA =1;           //开中断
    while(sign ==0)
    {
    P0 =0xc6;  //显示c 表示电动机发生正转
    P1 =0x01;  //给数码管位码端置高电平
    for(i =0;i <4;i ++)        //4 相
        {
        P2 =ctable[i];  //输出电动机正向转动表
        delay(500);        //改变这个参数可以调整电动机转速，数字越小，转速越大
        }
    }
    while(sign ==1)
    {
    P0 =0x86;  //显示E 表示电动机发生反转
    P1 =0x01;  //给数码管位码端置高电平
    for(i =0;i <4;i ++)        //4 相
        {
        P2 =etable[i];  //输出电动机反向转动表
        delay(500);        //改变这个参数可以调整电动机转速，数字越小，转速越大
        }
    }
    while(sign ==2)  //停止电动机转动
    {
    P0 =0x8e;  //显示F 表示电动机停止转动
    P1 =0x01;  //给数码管位码端置高电平
    }
    }
    void ISR_Key(void) interrupt 2 using 1
    {
    delay(500);
    if(!key)  //监测按键是否有按下
    {
    sign ++;         //按键按下触发一次
    if(sign ==3)
```

```
        sign = 0;
        }
    }
```

结合 2.3 节关于单片机开发环境 Keil C51 的学习，建立工程 dianji 文件，并将上述代码在 Keil 环境下进行编译，程序成功编译结果如图 15-6 所示。同时在创建工程路径 dianji 文件夹下生成一个扩展名为 .hex 文件，供下载软件将生成的 .hex 文件下载到单片机中。




图 15-6　程序编译成功界面

## 15.5　项目调试

针对上述对电动机转动系统硬件电路和软件程序设计，通过在研展科技 YZ200 单片机开发板上调试电动机转动电路，将 Keil 编译完成后产生的 .hex 文件结合 2.4 节 ISP 程序烧写知识，将 .hex 文件烧写到 51 单片机中。对电动机转动电路通电可以看到通过对按键的操作实现步进电动机的正、反转和停止功能，实现系统要求。

本项目实现了一个基于 51 单片机控制步进电动机实现电动机转动系统，用户根据按键操作控制步进电动机的运行，包括电动机的正、反转和停止。在实际应用中，可以扩充按键的使用，如通过按键操作增加电动机调速功能。如果实现步进电动机的加、减速控制，实际上是控制 CLK 的频率。加、减速控制就是不断改变定时器的初值，需要注意电动机转速时间足够短以及保证电动机控制速度的精确性，使得电动机设计上性能更趋向稳定。

# 第三部分　单片机综合案例实践篇

单片机综合案例实践部分是在单片机基础案例实践部分基础上对单片机应用掌握得更加深入与灵活。让读者从单片机知识学习的水平升华到利用单片机技术进行产品开发。通过单片机综合案例实践部分的学习，读者可以具备初步产品开发的能力，能够更加灵活的使用单片机进行各种电子系统产品设计。该部分由 9 个章节内容构成，9 个章节所涉及每个项目使用的源代码和硬件电路图请查询光盘资料。


第 16 章　家用温湿度测量播报系统设计

第 17 章　单片机实现智能充电器设计

第 18 章　无线遥控开关系统设计

第 19 章　融合物联感知与 GSM 的果园环境监测系统设计

第 20 章　单片机实现电子密码锁设计

第 21 章　红外遥控电动机转速系统设计

第 22 章　智能小车自动寻迹系统设计

第 23 章　红外遥控风扇控制系统设计

第 24 章　多功能微电脑模拟电子秤设计

# 第 16 章　家用温湿度测量播报系统设计

## 16.1　项目背景和设计意义

单片机诞生于 20 世纪 70 年代末，经历了 SCM、MCU、SoC 三大阶段。单片微型计算机（Single Chip Microcomputer，SCM）阶段，主要是寻求最佳单片形态嵌入式系统的最佳体系结构。“创新模式”获得成功，奠定了 SCM 与通用计算机完全不同的发展道路。微控制器（Micro Controller Unit，MCU）阶段，主要的技术发展方向是不断扩展满足嵌入式应用时，对象系统要求的各种外围电路与接口电路，突显其对象的智能化控制能力。单片机是嵌入式系统的独立发展之路，向 MCU 阶段发展的重要因素，就是寻求应用系统在芯片上的最大化解决。因此，专用单片机的发展自然形成了片上系统（System on Chip，SoC）化趋势。随着微电子技术、IC 设计、EDA 工具的发展，基于 SoC 的单片机应用系统设计会有较大的发展。因此，对单片机的理解可以从单片微型计算机、单片微控制器延伸到单片应用系统。

### 16.1.1　项目背景

8051 类单片机最早由 Intel 公司推出的 8051/31 类单片机，也是世界上用量最大的几种单片机之一。由于 Intel 公司在嵌入式应用方面将重点放在 286、386、奔腾等与 PC 类兼容的高档芯片的开发上，8051 类单片机主要由 Philips、三星、华邦等公司接产。这些公司都在保持与 8051 单片机兼容的基础上改善了 8051 许多特性（如时序特性）。提高了速度、降低了时钟频率，放宽了电源电压的动态范围，降低了产品价格。

在人类的生活环境中，温湿度扮演着极其重要的角色。特别是现在进入了数字化信息时代，对生活品质的要求也越来越高了。汽车、空调器等都已经家喻户晓了，它们都离不开温湿度等环境因素，因此也推进了温湿度传感器的发展。SHT10 温湿度传感器是利用 CMOSens 专利技术制成，确保产品有极高的可靠性和长期的稳定性。该温湿度传感器由一个电容式聚合体测湿元件和一个能隙式测温元件组成，并与一个 14 位 A-D 转换器以及一个 2wire 数字接口在芯片中无缝结合，使产品具有体积小、相应速度快、接口简单、性价比高等特点。适合远距离测温湿、控温湿，不需要进行非线性校准，外围电路简单。它是目前在国内外应用最为普遍的一种集成温湿度传感器，典型产品有 SHT10、SHT11、SHT75 等。西门子公司推出的 QFM2160、QFM3160 温湿度传感器，也是目前国内普遍使用的一种温湿度传感器，是由一个 PT1000 温度敏感元件和一个电容式湿度传感器组成，其工作电压 24V，功耗 0. 5V · A，湿度范围 10% ~90% RH。该传感器只有用于高湿、高温、酸性和弱碱性等比较恶劣的环境，测量稳定性较好。智能温湿度传感器内部都包含温度传感器、湿度传感器、A-D 转换器、信号处理器、存储器（或寄存器）和接口电路。有的产品还带多路选择器、中央控制器（CPU）、随机存取存储器（RAM）和只读存储器（ROM）。智能温湿度传感器的特点是能输出温湿度数据及相关的温湿度控制量，适配各种微控制器（MCU）；并且

它是在硬件的基础上通过软件来实现测试功能的，其智能化程度也取决于软件的开发水平。

### 16.1.2　项目设计意义

随着社会的发展，人们对于温湿度的控制和精确度也越来越高，在当今高速发展的社会中，对温湿度系统需求程度非常高，由于不同方面的温湿度需求，不同的应用范围，不同的传输实现方式等，就有了不同的温湿度测量系统的出现。目前国内外通过 AD 采样热敏电阻电压测量温度和电解质湿度的方法很普遍。而采用这种方法很难进行高精度的温度和湿度测量，仅能大致测量湿度值，不能满足现代工业对温湿度精度的要求。这种方法对单片机控制单元来说程序量较大，增加控制难度，而且其外围电路复杂，增加日后检修工作量。然而采用 MCU ＋SHT10 方法能满足工业对温湿度精度的要求，而且其外围电路也比较简单。

Sensirion 传感器公司生产的 SHT10 新型集成数字式温湿度传感器，利用 CMOSens 专利技术将温湿度传感器，A-D 转换器及数字接口无缝结合，具有体积小、响应速度快、接口简单、性价比高等特点。SHT10 温湿度传感器的温湿度范围较大，即温度测量范围可以达到 －40 ~123.8℃，在温度范围内误差仅 ±0.5℃；湿度测量范围可以达到 0 ~100% RH，在湿度范围内误差仅 ±4.5% RH。SHT10 可直接将温湿度转化成串行数字信号，因此特别适合和单片机配合使用，直接读取温湿度数据。目前 SHT10 数字温湿度传感器已经广泛应用于恒温室、粮库、游泳池及其他各种温湿度测控系统中。

相比之下，传统的温湿度检测系统采用热敏电阻等温度敏感元件和电解质湿度传感器，热敏电阻和电解质湿度传感器成本低，但需要后续信号调理、A-D 转换处理电路才能将温湿度信号转换成数字信号，不但电路复杂，而且热敏电阻和电解质湿度传感器的可靠性相对较差，测量温湿度的精度差，很难保证热敏电阻和电解质湿度传感器的一致性和线性，在应用中需要很好的解决引线误差补偿问题、共模干扰问题和放大电路零点漂移误差等技术问题。

## 16.2　项目方案论证和方案选择

### 16.2.1　项目方案论证

（1）方案一　由于本设计是温湿度测量电路，可以使用热敏电阻和电解质湿度传感器之类的器件，利用其感温和感湿效应，随被测温湿度变化的值采集过来，进行 A-D 转换后，使用单片机进行数据处理，在显示电路上，可以将被测温湿度显示出来，这种设计需要用到 A-D 转换电路，感温电路和感湿电路，比较麻烦。

（2）方案二　鉴于方案一外围电路设计的复杂性，考虑采用数字式温湿度传感器，该传感器具有 A-D 转换器及数字接口无缝结合、体积小、响应速度快、接口简单、性价比高等特点，同时可以很容易直接读取被测温湿度值，进行转换，就可以满足设计要求。

### 16.2.2　设计方案选择

通过综合应用所学电子技术、参数检测技术、单片机技术设计并实现一个较为实用的家用温湿度测量播报系统，该家用温湿度测量播报系统使用温湿度传感器 SHT10 对环境温湿度进行检测，并能利用语音芯片 ISD1420 进行语音播报，同时在 LCD1602 上显示相应的数

值；按照项目要求设计该装置的单片机最小系统、输入/输出接口电路、参数检测变换电路、语音播报电路和电源电路等，绘制系统各部分电路原理图，利用 PC/MCS-51 单片机联机开发系统编制和调试系统应用软件，并进行软硬件联调。

## 16.3　家用温湿度测量播报系统原理及功能

### 16.3.1　家用温湿度测量播报系统工作原理

本次设计的课题是家用温湿度测量播报系统的设计与实现。主要实现的系统功能是对环境温湿度的监测，在主机上进行温湿度数据显示，并将从传感器采集的信号实时地传给单片机，单片机可以知道是否存在异常情况。本次设计是通过单片机采集传感器信息，按照系统设计功能要求，系统由 4 个模块组成：主控制器、温湿度测量电路、显示电路和语音播报电路。主控制器采用 AT89S52，测温电路采用 Sensirion 传感器公司生产的 SHT10 新型集成数字式温湿度传感器，显示采用 1602 液晶显示，语音电路采用控制芯片 ISD1420 实现。单片机控制 SHT10 完成温湿度转换的思路为每一次读写之前都要对 SHT10 进行检测，DATA 处于低电平，连接成功后发送一条 ROM 指令，这样才能对 SHT10 进行预定的操作。首先要求启动数据传输，然后进行传感器复位，释放 DATA。SHT10 收到信号后，要求 SHT10 写数据，最后读数据，单片机收到此信号表示采集成功，然后显示屏上显示温度。SHT10 采集数据分 16 位，前 8 位为整数位，后 8 位为小数位，采集过程分为三个阶段：MSB 数据采集阶段、LSB 数据采集阶段、Checksum 和计算阶段。LCD 的应用则能更直观地对温湿度进行监控显示，绘制系统各部分电路原理图。利用 PC/MCS-51 单片机联机开发系统编制和调试系统应用软件，并进行软硬件联调。

设计方案主控制芯片采用 AT89S52 单片机，系统电路结构由单片机最小系统，温湿度采集电路，显示电路，语音播报电路，按键控制电路等组成，其结构框图如图 16-1 所示。

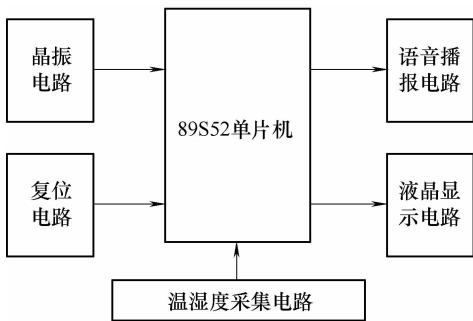



图 16-1　家用温湿度语音播报系统结构框图

### 16.3.2　家用温湿度测量播报系统功能分析

根据任务要求，选用 AT89S52 单片机为核心，组成温湿度语音播报电路。系统主要有 AT89S52 单片机、按键电路、显示电路、传感器电路、语音电路、电源电路等构成。通过按下不同的按键，单片机接收到不同的指令，控制相应的显示电路和语音电路。电源电路提供各部分模块工作电压。其中各部分功能模块实现要求如下：

**1. 温湿度采集与显示**

通过使用 LCD1602 液晶显示器对温度和湿度进行显示，利用单片机的 P2.5 和 P2.6 引脚对温湿度传感器 SHT10 进行温湿度的采集、转换，并通过 P0 口传送到 LCD 的 D0 ~ D7 上，显示当前温湿度值。

**2. 语音播报**

系统设计采用 ISD1420 语音芯片，其芯片内部集成电路由振荡电路、语音存储单元、前置放大器、自动增益控制电路、抗干扰滤波器、输出放大器组成。首先把 MIC 和 MIC REF 连接传声器，通过锁存开关触发录音功能，通过程序把要录的声音分地址段预先录到 ISD1420 里；电路工作时，语音芯片会接收单片机转送过来的温湿度，通过按键操作实现温湿度播报。

**3. 按键操作**

按键模块是通过对按键动作实现温湿度语音播报的功能。当按下 S1 键，实现温度语音播报，当按下 S2 键，实现湿度语音播报。

## 16.4　家用温湿度测量播报系统硬件电路设计

传统的温湿度测量系统不能同时测量温湿度，要分别通过有热电偶、热电阻和电解质湿度传感器实现温湿度的测量，而热电偶和热电阻测出的一半都是电压，在转换成对应的温度，同样湿度的测量也不能实现数字化，都需要比较多的外部硬件支持，硬件电路复杂，软件调试复杂，制作成本高。而数字温湿度测量播报系统研究均力求成本上更低，效果更好，应用更广泛并力图在数字化、无线化、集成化方面取得突破。具体表现为

①稳定，可靠：如探测器需可抗 RFI/EMI、防雷电等，以适应恶劣气候。

②多样的功能：如探测器可调频、防遮挡、防喷盖、防破坏等。

③精美、小巧的外观，以符合品味日益提高的室内装潢需求。

④智能化的设计，方便地设/撤防，人性化的操作界面。

为了达到以上的要求，故本次设计采用的传感器是由 Sensirion 传感器公司生产的 SHT10 新型集成数字式温湿度传感器，利用 CMOSens 专利技术将温湿度传感器，A-D 转换器及数字接口无缝结合，可以直接读出被测温湿度值，而且采用 4 线制与单片机相连，减少了外部硬件电路，具有低成本和易使用的特点。SHT10 温湿度传感器具有体积小、响应速度快、接口简单、性价比高等特点。

### 16.4.1　单片机最小系统模块设计

单片机最小系统主要由单片机 AT89S52、晶振电路和复位电路组成，如图 16-2 所示。其中单片机包括以下部分：

（1）中央处理器（CPU）　单片机的核心，完成运算和控制功能，AT89S52 单片机的 CPU 能处理 8 位二进制数或代码。

（2）内部数据存储器（内部 RAM）　AT89S52 芯片中有 128 个 RAM 单元，用于存放可读写的数据，简称内部 RAM。

（3）内部程序存储器（内部 ROM）　AT89S52 芯片内部有 8KB 掩膜 ROM，用于存放程序或表格，因此称为程序存储器，简称内部 ROM。

（4）定时/计数器　89S52 共有两个可编程的 16 位定时/计数器，以实现定时或计数为功能。

（5）并行 I/O 口　单片机共有 4 个可编程的 8 位的 I/O 口（P0，P1，P2，P3），以实现

数据的并行输入/输出。

（6）串行口　单片机有一个全双工的串行 I/O 口，以实现单片机和其他设备之间的串行数据传送。

（7）中断控制系统　AT89S52 单片机具有 5 个中断源，两个中断优先级的中断控制系统，以满足控制应用的需要。

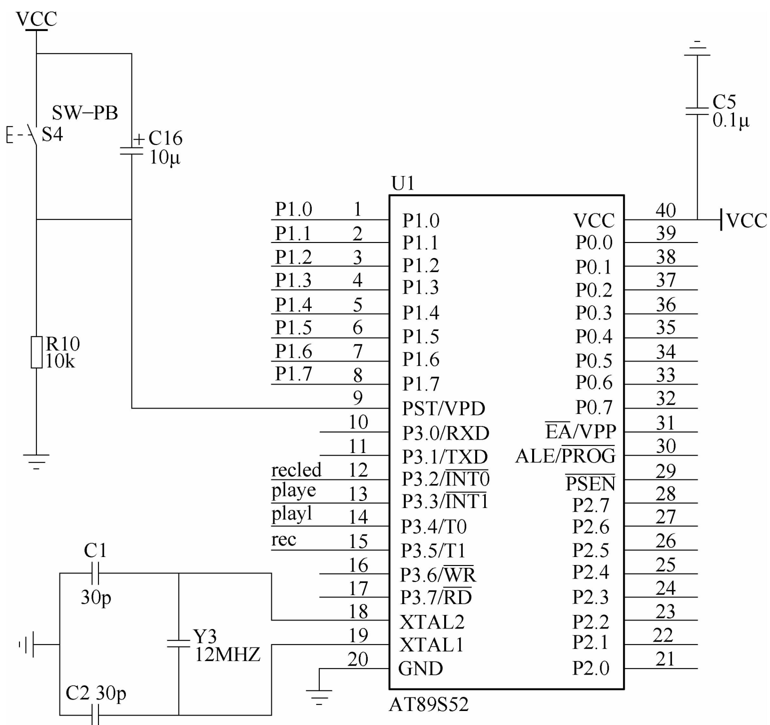



图 16-2　单片机最小系统电路图

晶振的作用是为系统提供基本的时钟信号，石英晶体和微调电容需外接。时钟电路为单片机产生时钟脉冲序列。系统允许的晶振频率为 2 ~ 12MHz。而本设计的最小系统使用 12MHz 的晶振，以提供系统所需的时钟频率。

系统复位是任何微机系统执行的第一步，使整个控制芯片回到默认的硬件状态下。本系统单片机的复位是由 RST 引脚来控制的，此引脚与高电平相接超过 24 个振荡周期后，AT89S52 进入芯片内部复位状态，而且一直在此状态下等待，直到 RST 引脚转为低电平。

## 16.4.2　温湿采集模块设计

SHT10 数字温湿度传感器属于主动方式，由 Sensirion 传感器公司生产。由一个电容式聚合体测湿元件和一个能隙式测温元件组成，并与一个 14 位 A-D 转换器以及一个 2wire 数字接口在芯片中无缝结合，使产品具有体积小、响应速度快、接口简单、性价比高等特点，外围电路简单。具有 4 引脚小体积封装形式；温度测量范围可以达到 −40 ~ 123.8℃，在温度范围内误差仅 ±0.5℃；湿度测量范围可以达到 0 ~ 100% RH，在湿度范围内误差仅 ±4.5% RH。被测温湿度用符号扩展的 16 位数字量方式串行输出。

SHT10 产品的特点如下：

①只要求两个端口即可实现通信。

②无需外部器件。

③可通过数据线供电，电压范围为 2.4 ~ 5.5V。

④零待机功耗。

⑤温湿度以 16 位数字量读出。

⑥温湿度同时测量、显示、播报。

SHT10 的外部引脚如图 16-3 所示。

VDD 电源引脚：SHT10 供电电压为 2.4 ~ 5.5V，传感器上电后，要等待 11ms，从休眠状态恢复。在此期间不发送任何指令。

SCK 串行时钟输入：SCK 引脚是单片机与 SHT10 之间通信的同步时钟。

DATA 串行数据：DATA 引脚是一个三态门，用于单片机与 SHT10 之间的数据传输。DATA 的状态在串行时钟 SCK 的下降沿之后发生改变，在 SCK 的上升沿有效。在数据传输期间，当 SCK 为高电平时，DATA 数据线上必须保持稳定状态。一般为避免数据发生冲突，单片机应该驱动 DATA 使其处于低电平状态，而外部接 1 个上拉电阻将信号拉到高电平。

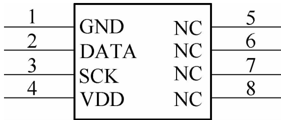



图 16-3　SHT10 引脚分布图

GND：接地端，一般 VDD 和 GND 之间可以增加一个 100nF 的电容器，用于去耦滤波。

SHT10 与单片机连接的电路如图 16-4 所示。

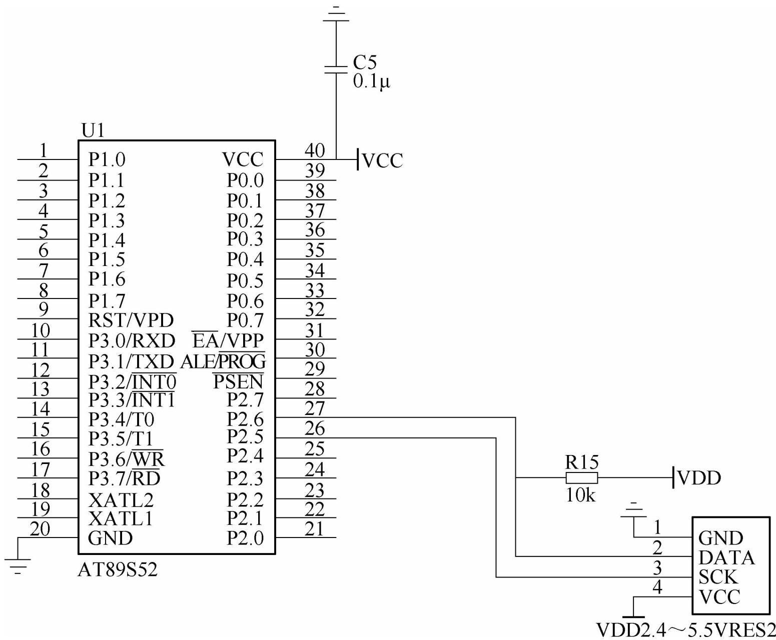



图 16-4　SHT10 与单片机连接图

SHT10 的引脚 4 与 VDD 相连，1 引脚与地相连，2 引脚和 3 引脚分别与单片机 P2.6 和 P2.5 引脚相连，电路工作时，通过程序 SHT10 可以把实时温湿度信息传送给单片机，单片

机对采集到的温湿度信息进行处理。

### 16.4.3 液晶显示模块设计

液晶显示模块是一种专门用于显示字母、数字、符号等点阵式 LCD，通过用 5×7 点阵图形来显示字符的液晶显示器，根据显示的容量可以分为 1 行 16 个字、两行 16 个字、两行 20 个字等，在本次设计中采用的是两行 16 个字的液晶显示模块。如图 16-5 所示。



图 16-5 液晶显示模块

液晶模块接口引脚描述见表 16-1。

**表 16-1 LCD 液晶显示屏接口引脚描述**

| 引脚号 | 符　号 | 状　态 | 功　能 |
|---|---|---|---|
| 1 | VSS | | 电源地 |
| 2 | VDD | | 5V 逻辑电源 |
| 3 | V0 | | 液晶驱动电源 |
| 4 | RS | 输入 | 寄存器选择 1：数据；0：指令 |
| 5 | R/W | 输入 | 读、写操作选择 1：读；0：写 |
| 6 | E | 输入 | 使能信号 |
| 7 | DB0 | 三态 | 数据总线（LSB） |
| 8 | DB1 | 三态 | 数据总线 |
| 9 | DB2 | 三态 | 数据总线 |
| 10 | DB3 | 三态 | 数据总线 |
| 11 | DB4 | 三态 | 数据总线 |
| 12 | DB5 | 三态 | 数据总线 |
| 13 | DB6 | 三态 | 数据总线 |
| 14 | DB7 | 三态 | 数据总线（MSB） |
| 15 | LEDA | 输入 | 背光 5V |
| 16 | LEDK | 输入 | 背光地 |

注：15、16 两引脚用于带背光模块，不带背光的模块这两个引脚悬空不接。

其 LCD 与单片机相连的电路如图 16-6 所示。

液晶显示电路接通时单片机读取 SHT10 的温湿度值，通过 P0.0 ~ P0.7 端口传送到 LCD 的 D0 ~ D7 端口上，P2.7 触发 E 为高电平、R/W = 1，并在 LCD 第一排显示当前温度，同时在 LCD 第二排显示当前湿度。对于最大和最小门限温湿度不作修改，按下温度播报按键或

湿度播报按键时，通过中断，调用子程序实现语音播报，播报完成，LCD 工作，显示当前温湿度。

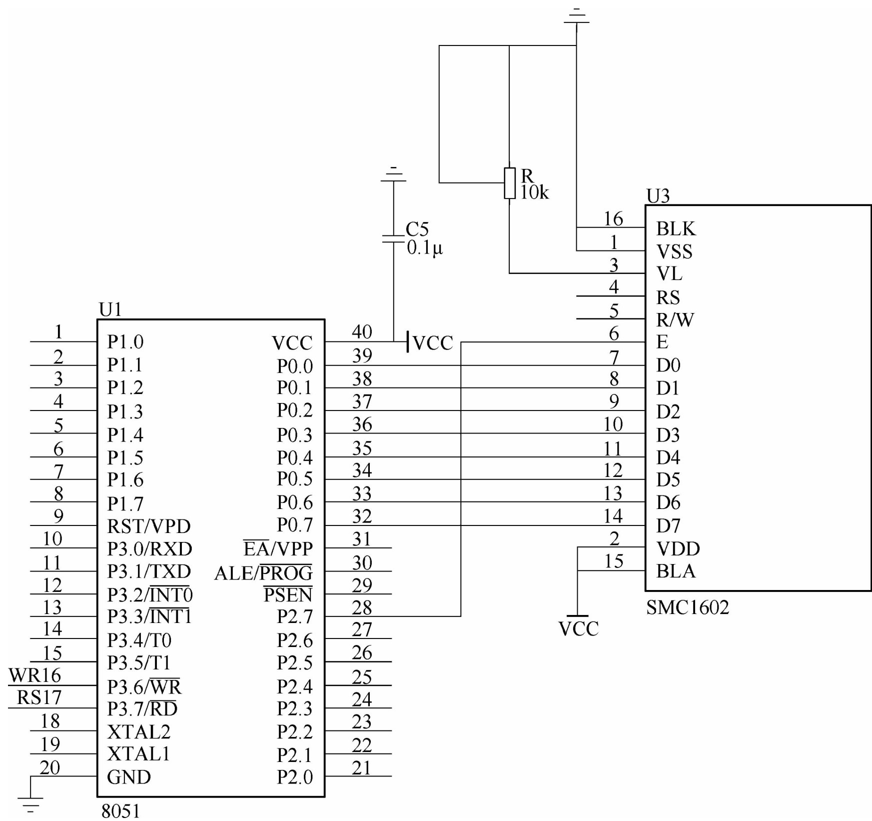



图 16-6　LCD1602 与单片机的连接图

### 16.4.4　语音播报模块设计

ISD1420 系列工作电压 5V，单片录放时间长，音质好，适用于移动电话及其他便携式电子产品中。芯片采用直接模拟存储技术（DAST TM），内含振荡器、语音存储单元、前置放大器、自动增益控制电路、抗干扰滤波器、音频输出放大器。一个最小的录放系统仅由一个传声器、一个扬声器、一个电源、两个按钮、少数电阻电容组成。芯片采用多电平直接模拟量存储技术，每个采样值直接存贮在片内闪存中，因此能够非常真实、自然地再现语音、音乐、音调和效果声。其中 ISD1420 语音芯片引脚说明如下：

A0 ~ A7：地址输入端，当 A6 和 A7 不全为高电平时，A0 ~ A7 为分段录音信息地址线，不同的地址对应不同的录音片断，A6 和 A7 全为高电平时，A0 ~ A5 用于选择操作模式。

MIC：传声器输入端，传声器输入信号通过电容交流耦合并传给片上预放大器，片上自动增益控制（AGC）电路控制预放大器的增益在 15 ~ 24dB 之间。耦合电容值和该端内阻（10kΩ）决定语音信号通频带下限频率。

MIC REF：传声器参考输入端，MIC REF 是预放大器的反相输入端，配合外电路可使片上预放大器具有较高的噪声抑制比和共模抑制比。

ANA IN：模拟信号输入端，对于传声器输入，ANA IN 引脚应通过外部电容与 ANA

OUT 引脚连接，若为外部输入信号，则要直接通过电容耦合到此端。耦合电容决定片上控制预放大器通频带的下限频率。

ANA OUT：预放大器的输出端，预放大器的电压增益取决于 AGC 电平，对于小信号输入电平，其增益最大为 24dB，对于强信号，增益较低。

AGC：自动增益控制端，AGC 动态地调整预放大器增益，使加至 MIC 输入端的非失真信号的范围扩展。内阻抗（5kΩ）和外部电容决定 AGC 的响应时间，外部电容和外部电阻的 RC 时间常数决定 AGC 的释放时间。

SP+、SP-：扬声器输出端，该端可直接驱动 16Ω 扬声器。可采用双端输出驱动扬声器，也可采用单端输出驱动扬声器，不过双端输出信号的功率是单端的 4 倍，单端输出需要该脚与扬声器之间串接 100μF 的交流耦合电容，录音期间该输出端保持高阻状态。

XCLK：外接时钟输入端，ISD1420 具有内部时钟，一旦接入外部时钟，内部时钟会自动失去作用，如果不用外部时钟该引脚应当接地，一般不推荐使用外部时钟，除非要求时钟信号特别精确。

RECLED：工作状态指示端，在录音或放音时该端输出低电平，可驱动一个 LED 来指示状态。

PLAYE：边沿触发放音控制端，该端输入一低脉冲，芯片即进入放音状，直至遇到信息结束标记（EOM）或到存储空间的末尾时回放过程结束，电路自动进入准备状态。回放过程中 PLAYE 变化不会影响回放过程。

PLAYL：电平触发放音控制端，该端电平变为低电平并保持，芯片进入放音状态，放音过程持续到该端电平由低变高或遇到信息结束标记（EOM），结束后电路进入准备状态。

REC：录音触发端，REC 一旦变为低电平，芯片就进入录音状态，REC 的权限优先于 PLAYE 和 PLAYL，在放音期间若遇 REC 接低电平时，放音就会立即停止并转入录音状态开始录音。录音期间 REC 应始终保持低电平，REC 变高或存储空间变满时录音过程结束，这时在录音截止的地方会记录一个信息结束标记（EOM）。

VCCD、VCCA：数字电源正端和模拟电源正端，为了减小片内噪声，芯片中模拟电路和数字电路在内部是分开的，应用时两个电源引脚应离电源尽可能的近，而且电源的去耦电容应离引脚越近越好。

VSSD、VSSA：数字地和模拟地。

将 ISD1420 引脚功能见表 16-2。

**表 16-2 ISD1420 引脚功能**

| 名　称 | 引　脚 | 功　能 | 名　称 | 引　脚 | 功　能 |
|---|---|---|---|---|---|
| A0～A5 | 1～6 | 地址 | ANA OUT | 21 | 模拟输出 |
| A6、A7 | 9、10 | 地址(MSB) | ANA IN | 20 | 模拟输入 |
| VCCD | 28 | 数字电路电源 | AGC | 19 | 自动增益控制 |
| VCCA | 16 | 模拟电路电源 | MIC | 17 | 传声器输入 |
| VSSD | 12 | 数字地 | MIC REF | 18 | 传声器参考输入 |
| VSSA | 13 | 模拟地 | PLAYE | 24 | 放音,边沿触发 |
| SP+、- | 14、15 | 扬声器输出+、- | REC | 27 | 录音 |
| XCLK | 26 | 外接定时器(可选) | RECLED | 25 | 发光二极管接口 |
| NC | 11 | 空脚 | PLAYL | 23 | 放音,电平触发 |

ISD1420 的工作模式主要分为操作模式和地址模式。

**1. 操作模式**

地址输入有双重功能，根据地址中的 A6，A7 的电平状态决定 A0 ~ A7 的功能。如果 A6，A7 有一个是低电平，A0 ~ A7 输入为地址位，作为起始地址用。地址位仅作为输入端，在操作过程中不能输出内部地址信息。根据 PLAYL、PLAYE 或 REC 的下降沿信号，地址输入被锁定。如果 A6，A7 同为高电平时，它们即为模式位。

使用操作模式有两点要注意：

①所有初始操作都是从 0 地址开始，0 地址是 1420 存储空间的起始端，以后的操作可根据模式的不同，而从不同的地址开始工作。当电路中录放音转换或进入省电状态时，地址计数器复位为 0。

②当 PLAYL 、PLAYE 或 REC 变为低电平，同时 A6，A7 为高电平时，执行对应操作模式。这种操作模式一直执行到下一个低电平控制输入信号出现为止，该地址/模式信号被取样并执行。

操作模式可以与微控制器一起使用，也可用硬件连线得到所需系统操作。

A0——信息检索，不知道每个信息的实际地址，A0 可使操作者快速检索每条信息，A0 每输入一个脉冲，可使得内部地址计数器跳到下一个信息。这种模式仅用于放音，通常与 A4 操作同时应用。

A1——删除标志可使录入的分段信息成为连续的信息，用 A1 可删除掉每段中间信息后的标志，仅在所有信息后留一个标志。当这个操作模式完成时，录入的所有信息就作为一个连续的信息放出。

A3——循环重放信息可使存于存储空间开始端的信息自动地连续重放。一条信息可以完全占满存储空间，那么循环就可以从头至尾进行工作，并由始至终反复重放。

A4——连续寻址，在正常操作中，当一个信息放出，遇到一个标志时，地址计数器会复位，A4 可防止地址计数器复位，使得信息连续不断地放出。

A2、A5——未用。

**2. 地址模式**

ISD1420 地址输入端具有双重功能，根据地址中的 A6、A7 的电平状态决定 A0 ~ A7 的功能。如果 A6、A7 有一个低电平，A0 ~ A7 输入为地址位，作为起始地址用，此时地址线仅作为输入端，在操作过程中不能输出内部地址信息。根据 PLAYE、PLAYL 或 REC 的下降沿信号，地址输入被锁定。如果 A6、A7 同为高电平时，它们即为模式位。

在这里只用到地址功能来分段控制，所以需要保证 A6、A7 不可同时为 1，可以用软件进行保护。

地址输入端 A0 ~ A7 有效值范围为 00000000 ~ 10011111，这表明最多可被划分为 160 个存储单元，可录放多达 160 段语音信息。由 A0 ~ A7 决定每段语音的起始地址，而起始地址又直接反映了录放的起始时间。其关系见公式：

$$TQ = 0.125\text{s} \times (128A7 + 64A6 + 32A5 + 16A4 + 8A3 + 4A2 + 2A1 + 0)$$

例如：

第一段语音从 0s 开始，地址设置为 00000000；

第二段语音从 2s 开始，地址设置为 00010000；

第三段语音从 5s 开始，地址设置为 00100000；

第四段语音从 12s 开始，地址设置为 01010000。

ISD1420 与单片机连接的电路如图 16-7 所示。

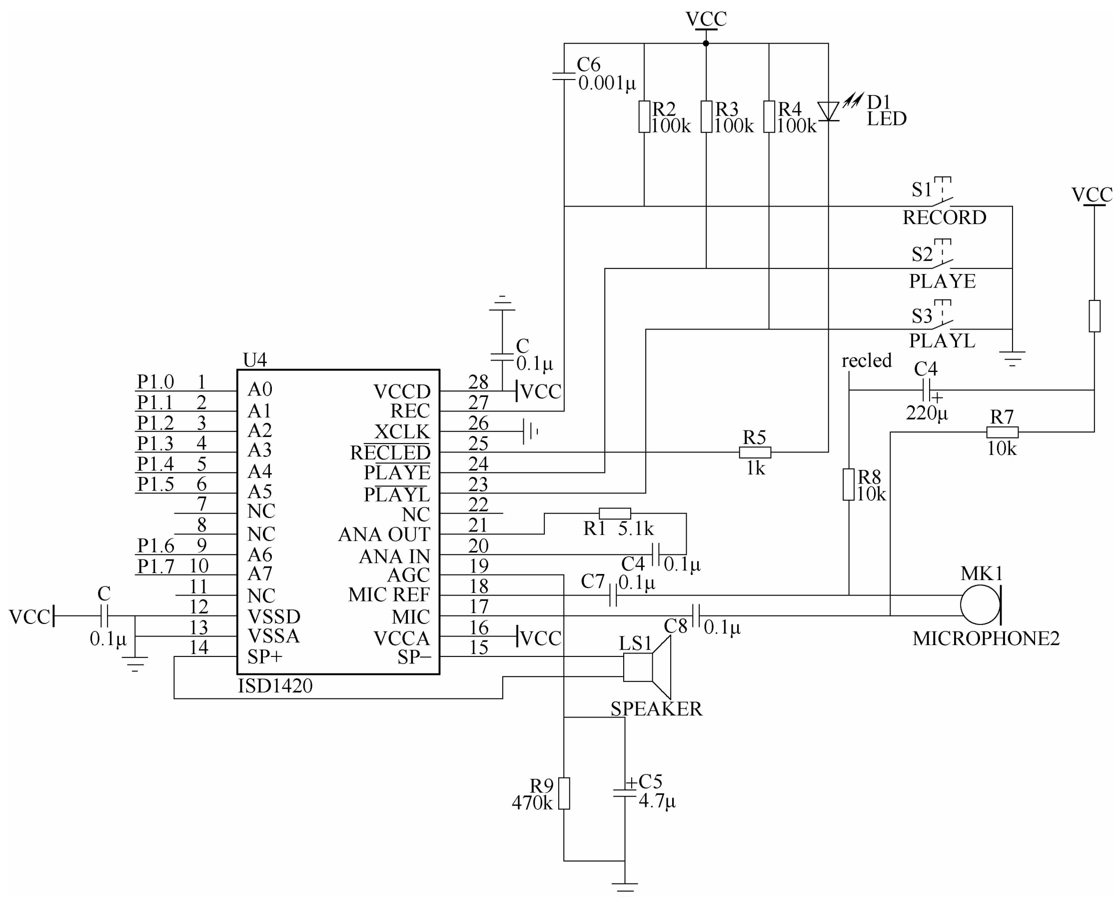



图 16-7 ISD1420 与单片机连接图

语音播报模块采用 ISD1420 语音芯片，由振荡电路、语音存储单元、前置放大器、自动增益控制电路、抗干扰滤波器、输出放大器组成。输入端接单片机的 P1 口，ISD1420 引脚 17 和引脚 18 分别接传声器和放大电路，语音芯片工作前，首先用锁存按键触发 REC，通过传声器对语音预先录制，完成后触发 PLAY 键进行语音播放。电路工作时，单片机先采集温度，通过 P1 口转送到 A0 ~ A7 口，根据预先录制的语音，触发播报按键进行语音播报。

## 16.5 家用温湿度测量播报系统的软件实现

C 语言是一种编译型程序设计语言，它兼顾了多种高级语言的特点，并具备汇编语言的

功能。用 C 语言来编写目标系统软件，会大大缩短开发周期，便于改进和扩充。用 C 语言进行 51 单片机程序设计是单片机开发与应用的必然趋势。单片机程序设计应该以 C 语言为主，以汇编语言为辅，采用 C 语言不必对单片机和硬件接口的结构有很深入的了解，编译器可以自动完成变量存储单元的分配，编程者可以专注于应用软件部分的设计，大大加快软件的开发速度。采用 C 语言很容易进行单片机的程序移植工作，有利于产品中单片机的重新选型。

现在的单片机仿真器普遍支持 C 语言程序的调试，为单片机编程使用 C 语言提供了便利的条件。C 语言模块化程序结构可以使程序模块供用户分享，不断丰富。采用 C 语言可以针对单片机常用的接口芯片编制通用的驱动函数，方便用户使用。

单片机的 C 语言采用 C51 编译器常用的 Keil C51。由 C51 产生的目标代码，其运算速度高，所需存储空间小，符合 C 语言的 ANSI 标准，生成的代码遵循 Intel 目标文件格式，而且可与 A51 汇编语言或 PL/M51 语言目标代码混合使用。

应用 C51 编程具有以下优点：

（1） C51 管理内部寄存器和存储器的分配，编程时无须考虑不同存储器的寻址和数据类型等细节问题。

（2） 程序有若干函数组成，具有良好的模块化结构。

（3） 有丰富的子程序库可直接引用，从而大大减少用户编程的工作量。

（4） C 语言和汇编语言可以交叉使用，汇编语言程序代码短，运行速度快，但复杂运算编程耗时。用汇编语言编写与硬件有关的部分程序，用 C 语言编写与硬件无关的运算部分程序，充分发挥两种语言的长处，提高开发效率。

### 16.5.1　单片机控制主程序软件设计

主程序的主要功能是初始化并负责温湿度的读出、处理计算及显示。温湿度测量每 1s 进行一次，包括液晶显示模块功能初始化函数说明（如字符设置为 8 位数据接口，两行显示以及 7 点阵显示，开显示，清屏等），键盘扫描模块，定义完毕后通过对按键 S1 和 S2 的操作，实现对系统的温湿度语音播报功能。经过数据换算，由于字符型液晶显示模块内有固定的字模库，就不需要自己进行字模转换。数据通过 DB0 ~ DB7 传送至 LCD 以及光标的不断移位而得到显示。

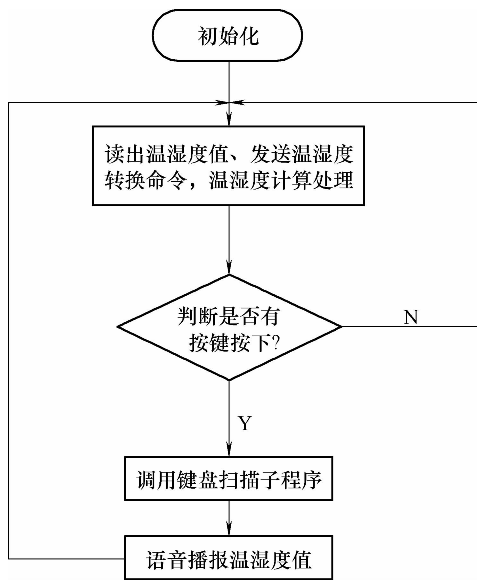



图 16-8　系统主程序流程图

其流程图如图 16-8 所示。

主程序关键代码如下：

```
void main(void)
{
    value humi_val,temp_val;                //定义温湿度共用体变量
    unsigned char error,checksum;           //定义 error,checkusm
```

```
    unsigned int wendu,shidu;                    //定义温湿度值
    LCD_init();                                  //LCD1602初始化
    s_connectionreset();                         //温湿度采集
    LCD_disp_str(0,1,"TE");                      //温度液晶显示地址
    LCD_disp_str(0,2,"RH");                      //湿度液晶显示地址
    LCD_disp_str(2,1,"TTT. TC");                 //温度液晶显示字符
    LCD_disp_str(2,2,"RRR. R%");                 //湿度液晶显示字符
    delay_n10us(20000);                          //延时0.2s
    while(1)
    {
        scan_key();          //调用按键扫描(按键扫描中调用语音播报子函数)
        error =0;
        error += s_measure((unsigned char *) &humi_val. i,&checksum,HUMI);
                                                     //计算湿度值
        error += s_measure((unsigned char *) &temp_val. i,&checksum,TEMP);
                                                     //计算温度值
        ...
            delay_n10us(80000);                      //延时约0.8s
    }
}
```

### 16.5.2　温湿采集程序设计

温度转换命令子程序主要是单片机接收检测信号开始，在本程序设计中首先定了union两个共用体，一个是整型i；另一个是单精度f。同时采用各种算法转换，等待转换温湿度值的数据形式，完成数据类型转换。

温湿度采集关键代码如下：

```
    typedef union                                  //定义自定义类型typedef
    {
        unsigned int i;                            //定义共用体整型"i"
        float f;                                   //定义共用体单精度字符"f"
    }
    value;                                         //定义结构类型value,value=typedef union
        ...
    for (i =0;i <65535;i ++)
    if(DATA ==0) break;
    if(DATA) error + =1;                           //接收SHT10检测信号
     *(p_value) =s_read_byte(ACK);                 //读取温湿度值
     *(p_value +1) =s_read_byte(ACK);              //读取温湿度值
     *p_checksum  =s_read_byte(noACK);             //读取温湿度值
```

```
return error;
     ...
if(error!=0)
     s_connectionreset();                    //温湿度数据采集
else
{
     humi_val.f=(float)humi_val.i;           //湿度数据类型转换
     temp_val.f=(float)temp_val.i;           //温度数据类型转换
     calc_dht90(&humi_val.f,&temp_val.f);
}
```

### 16.5.3　LCD 显示程序设计

单片机将显示的数据通过 D0 ~ D7 口传送到 LCD，触发 E 为高电平、R/W =1，同时显示 SHT10 温湿度，其程序流程图如图 16-9 所示。

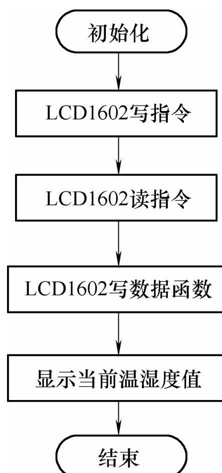



图 16-9　LCD 显示温湿度流程图

温湿度数据的计算处理方法：从 SHT10 读取的二进制数值必须先转换成十进制数值，才能用于字符的显示。利用温湿度值的除和余的算法，实现采集温湿度值的十进制转换，因此 LCD1602 显示的就是实际的十进制温湿度值。

```
LCD_disp_str(0,1,"TE");                      //温度液晶显示地址
LCD_disp_str(0,2,"RH");                      //湿度液晶显示地址
LCD_disp_str(2,1,"TTT.TC");                  //温度液晶显示字符
LCD_disp_str(2,2,"RRR.R%");                  //湿度液晶显示字符
...
wendu=10*temp_val.f;
LCD_disp_char(2,1,wendu/1000+'0');           //显示温度百位
LCD_disp_char(3,1,(wendu%1000)/100+'0');     //显示温度十位
```

```
LCD_disp_char(4,1,(wendu%100)/10 +'0');      //显示温度个位
LCD_disp_char(6,1,(wendu%10) +'0');          //显示温度小数点后第一位
shidu =10 * humi_val. f;
LCD_disp_char(2,2,shidu/1000 +'0');          //显示湿度百位
LCD_disp_char(3,2,(shidu%1000)/100 +'0');    //显示湿度十位
LCD_disp_char(4,2,(shidu%100)/10 +'0');      //显示湿度个位
LCD_disp_char(6,2,(shidu%10) +'0');          //显示湿度小数点后第一位
```

### 16.5.4　语音播报程序设计

利用不同按键的操作控制温湿度的分别播报。通过调用不同的语音地址，实现单片机的温湿度语音播报功能，其流程图如图 16-10 所示。

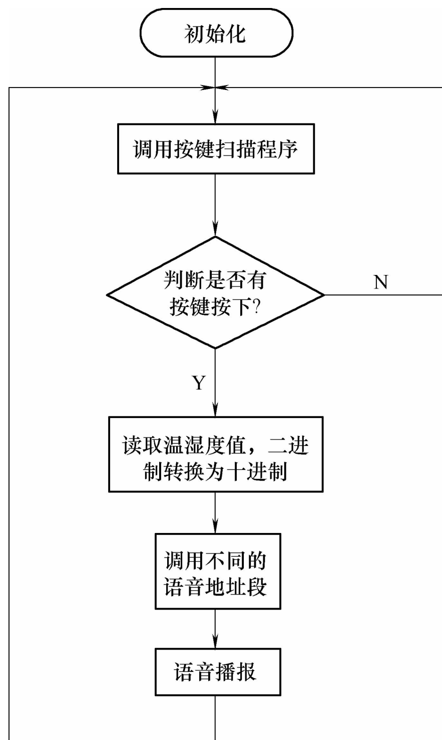



图 16-10　语音播报流程图

温湿度播报调用的语音地址，都是通过录音程序在 ISD1420 的 0x00-0xff 段地址进行分段录音，每段存储单一的音频文件，播报程序通过对各个地址段的调用，实现温湿度语音的连续实时播报。

```
void speak_shidu(unsigned int shidu)
{
    int digit3,digit2;                 //定义两个整型变量
    digit3 =shidu / 10;                //digit3 等于值除 10
    digit2 =shidu % 10;                //digit3 等于值余 10
```

```
        if( digit3 ==0)
        {
            speak_isd1420( speak_world[ digit2 ] ) ;      //调用语音播报程序，播报个位
            speak_isd1420( speak_world[ 12 ] ) ;
            //调用语音播报程序,播报“百分之”
        }
        else
        {
            speak_isd1420( speak_world[ digit3 ] ) ;      //调用语音播报程序，播报十位
            speak_isd1420( speak_world[ 10 ] ) ;          //调用语音播报程序，播报“十”
        if( digit2 !=0)
            speak_isd1420( speak_world[ digit2 ] ) ;      //调用语音播报程序,播报个位
            speak_isd1420( speak_world[ 12 ] ) ;
          //调用语音播报程序，播报“百分之”
        }
    }
```

## 16.6　系统调试

### 16.6.1　软件调试

本设计中源程序的编写与调试是在 Keil C51 集成环境中进行的。首先建立一个新的工程，单击菜单 project，选择 new project，然后选择要保存的路径，输入工程文件的名字，如保存到 keil 目录里，工程文件的名字为 hejiajian，然后单击保存按钮。这时会弹出一个对话框，要求你选择单片机的型号，你可以根据使用的单片机来选择，keil C51 几乎支持所有的 51 类型的单片机，如果设计的是 AT89S52，可以选择 Amtel -> AT89S52，选择 89S52 之后，右边一栏是对这个单片机基本的说明，然后单击确定按钮。这时要新建一个源程序文件，建立一个汇编或 c 文件，如果你已经有源程序文件，可以忽略这一步。单击菜单 File -> New 命令，输入一个简单的程序，选择菜单 File→SAVE 命令，选择你要保存的路径，在文件名里输入文件名，注意一定要输入扩展名，由于是 c 程序文件，扩展名为 . c。保存为 hejiajian. c 的名字，（也可以保存为其他名字，如 learn. c 等），单击保存按钮。单击 Target 1 前面的 + 号，展开里面的内容 Source Group1，用右键单击 Sourece Group 1（注意用鼠标的右键，而不是左键），将弹出一个菜单，选择 Add Files to Guoup 'Source Group 1 '，文件类型选择 C Source file（ *. c）。选择刚才的文件 test. c，最后单击 Add 按钮。单击 add 按钮之后，窗口不会消失，（如果要添加多个文件，可以不断添加），添加完毕此时再单击 Close 关闭该窗口，这时在 Source Group 1 里就有 hejiajian. c 文件。

程序在 Keil 软件下编译成功，如图 16-11 所示。

在 Keil 软件下编译成功后，生成 . hex 文件，再用 ISP 和单片机程序烧写器将图 16-11 编译成功的程序下载到单片机 AT89S52 中，其下载成功的界面如图 16-12 所示。

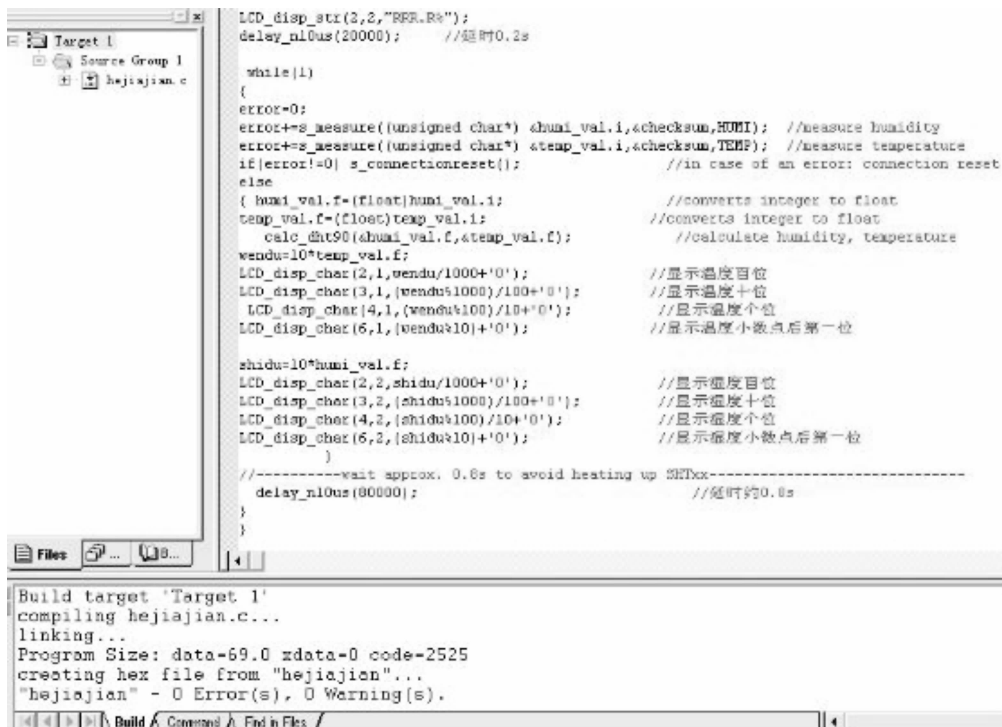



图 16-11　程序调试图




图 16-12　程序下载图

## 16.6.2　实物调试中遇到问题

图 16-13 为调试成功的家用温湿测量播报系统实物图，图中的状态是电路通电后，按温度播报键（4 个按键中的第 3 个）一下，播报当前的温度值；按下湿度播报键（4 个按键中的第 4 个），播报当前的湿度值。LCD 第一行显示 SHT10 传送过来的当前温度 19.8℃，第二行显示当前的湿度 50.3%。

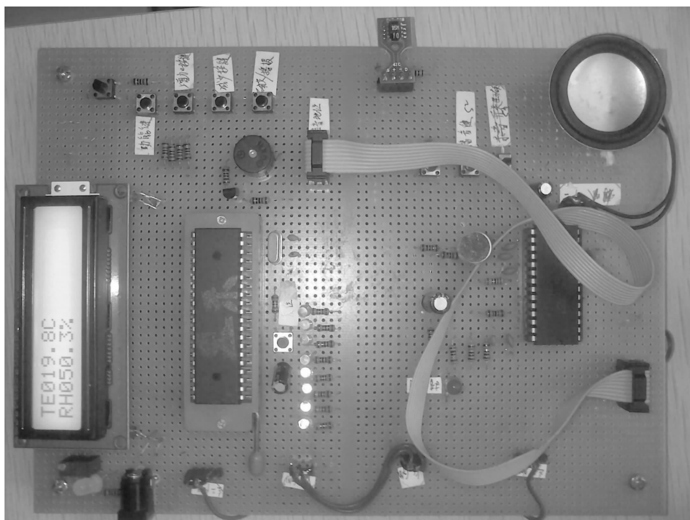



图 16-13　家用温湿测量播报系统实物图

本设计在系统调试中遇到很多问题：

（1）在调试温湿度传感器 SHT10 时，显示不了温湿度，经发现是因为 SHT10 是贴片，焊接的时候在温度达到 350°C 的状态下不超过 5s。由于贴片焊接不熟练，导致传感器烧毁。重新购买后完成。

（2）在液晶显示器显示温湿度时，LCD1602 显示不了温湿度，经过调试是由于 LCD1602 使用的排针和排插的原因，原来是地址线排插里面的金属片丢失，导致数据无法正常传输，修正后解决。

（3）用万用表测量各引脚导通情况，发现 ISD1420 的地址线和单片机相连处出现虚焊现象，接触不良，导致语音芯片录放音出现地址混乱。对于虚焊处重新焊接，测量无误。

（4）调试语音芯片时发现不能录音和放音，经检查时 ISD1420 语音芯片自身损坏造成，重新换过后调成正常。

（5）语音芯片的录音过程由于用到多段地址，需要分段录音。由于地址段之间的时间段，分段录音难以掌握，时常出现下端音频覆盖上段音频，导致语音播报时调用地址段不完整。经过地址段的时间计算，并且多重尝试，录制语音成功。

## 16.7　总结

本设计实现一个较为实用的家用温湿度语音播报系统，该家用温湿度语音播报系统使用温湿度传感器 SHT10 对环境温湿度进行检测，并能利用语音芯片 ISD1420 进行语音报数同时在 LCD 上显示。设计的家用温湿度语音播报系统能够实时监测温湿度等环境因素，系统设计成本低，显示效果好，实用性可靠性强，并具有操作简单等特点。其中采用 SHT10 作为温湿度传感器，比热敏电阻和电解质湿度传感器具有减小外界的干扰，提高测量的精度，另外在显示温湿度的时候同时具有语音播报功能，更具人性化、更加便捷。项目也可以在功能上进一步改善，如在液晶显示上，可以使用 LCD12864，做到信息的中文显示，达到更加直观的显示效果；同样可以增加时间、温湿度门限报警系统，提供更多的辅助功能。

# 附件：设计的电路原理图

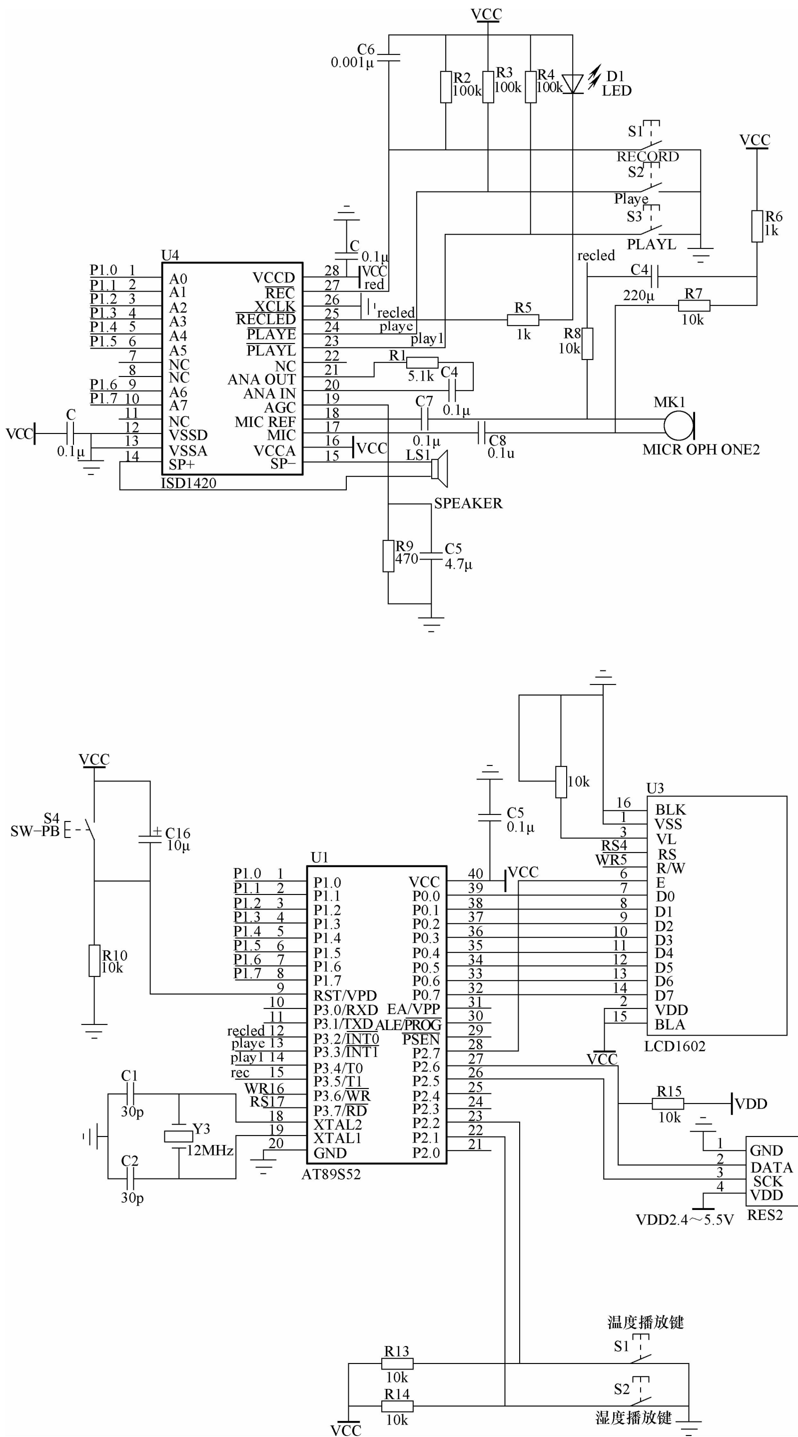


图 16-14　家用温湿测量播报系统电路原理图

# 第 17 章　单片机实现智能充电器设计

伴随着半导体元件、大规模集成电路的快速发展以及人们生活水平的不断提高，各种便携式电动剃须刀、MP3、便携式计算机、摄像机、电子计算器、移动电话等电器广泛应用，使得各种干电池的需求量迅速增加。但是一次性的普通碱性电池使用寿命较短，特别是对诸如相机一类功耗较大的小型家用电器来说，一般 1 ~ 2h 就需更换电池。因此人们普遍希望用可充电的干电池来代替一次性的干电池，尽管可充电电池成本较高，一次性投资较大，但由于可反复充电使用，其总成本相对还是比较低的。普通一次性碱性电池如果使用得当，并且采用适当的充电方法也可以反复使用，充放电循环可达 30 次以上，甚至可以用上几年。若采用可充电的电池（如镍福电池），虽然一次性投入资金较高，但使用寿命很长，并且可以反复充放电 1000 次以上。而不管是一次性碱性电池还是可充电电池都可以通过适当的充电器来延长其使用寿命。因此充电器的开发与应用，对节约原材料、节约能源、减少环境污染具有重要意义。

## 17.1　项目背景和设计意义

在人们的日常工作和生活中，充电器的使用是越来越广泛。从 MP3 到数码相机，从手机到便携式计算机，几乎所有用到电池的电器设备都需要用到充电器。充电器为人们外出的旅行和出差提供了极大地方便。

### 17.1.1　项目背景

充电器是采用电子半导体器件，将电压和频率固定不变的交流电变换为直流电的一种静止变流装置。在以蓄电池为工作电源或备用电源的用电场合，充电器具有广泛的应用前景。充电器有很多，如铅酸蓄电池充电器、阀控密封铅酸蓄电池的测试与监测、镉镍电池充电器、镍氢电池充电器、锂离子电池充电器、便携式电子设备锂离子电池充电器、锂离子电池保护电路多功能充电器、电动车蓄电池充电器、车充等。

充电器的发展经历了三个阶段：

（1）限流限压式充电器　最原始的就是限压式充电，然后过渡到限流限压式充电，它的使用方式就是浅充浅放，但这种充电模式的效果较差。

（2）恒流限压式充电器　这是充电器发展的第二阶段，这种模式的充电器占据了充电器市场近半个世纪。首先以恒电流充电至预定的电压值，然后改为恒电压完成剩余的充电。一般两阶段之间的转换电压就是第二阶段的恒电压。这种充电器充电电流总是低于电池的可接受能力，使得充电效率低下，大大降低了电池的使用寿命。

（3）自适应智能充电器　随着大规模集成 IC 的出现，充电器设备进入了一个全新的自适应、智能阶段，即称为第三代充电器。自适应充电器遵循各类电池的充放电规律进行充放电，并且具有温度的补偿功能。充电系统是由特殊功能的单片机控制，通过不断地检测系统

参数，并且按照模糊推理算法不断地调整充电参数。同一充电器可适应不同种类电池的充电，充电器自适应调整自己的输出电流，无需人工选择，同时避免了操作失误。

由于充电器大多采用大电流的快速充电方法，在电池充满后如果不及时停止便会使电池发烫。过度的充电会严重损害电池的寿命。一些低成本的充电器常常为了防止过充，一般充电到 90% 就停止大电流快充，而采用小电流涓流补充充电。

### 17.1.2　设计意义

手机电池的使用寿命和单次使用时间以及充电过程密切相关。锂电池是手机最常用的一种电池，它具有较高的能量重量比、能量体积比、具有记忆效应，可重复充电多次，使用寿命较长，价格也越来越低。锂电池对于充电器的要求比较苛刻，需要保护电路。为了有效利用电池容量，需将锂电池充电至最大电压，但是过电压充电会造成电池损坏，这就要求较高的控制精度。另外，对于电压过低的电池需要进行预充，充电器最好带有热保护和时间保护，为电池提供附加保护。

一部分的充电器不但能在很短时间内将电量充足，而且还可以对电池起到一定的维护作用，修复由于使用不当造成的记忆效应，即容量下降（电池活性衰退）现象。设计比较科学的充电器往往采用专用充电器控制芯片配合单片机控制的方式。专用充电芯片具有业界公认较好的检测，可以检测出电池充电饱和时发出的电压变化信号，比较精确地结束充电工作，并且通过单片机对这些芯片的控制，可以实现充电过程的智能化。例如在充电后增加及时关断电源、蜂鸣报警和液晶显示等功能。充电器的智能化可以缩短充电的时间，同时能够维护电池，延长电池的使用寿命。

单片机在充电器领域也有着广泛的应用，利用它的处理控制能力可以实现充电器的智能化。充电器种类繁多，但从严格意义上讲，只有单片机参与处理和控制的充电器才能称为智能充电器。51 系列的单片机也是当前使用最为广泛的 8 位单片机系列，其丰富的开发资源和较低的开发成本，使 51 系列单片机现在以至于将来都仍会有强大的生命力。在众多的 51 系列单片机中，AT89 系列单片机在我国得到了极其广泛的应用。AT89 系列单片机是美国 Atmel 公司的产品。其片内程序存储器采用闪存技术，编程/擦写速度快（擦除时间约 10ms，4K 编程约 3s），并可反复编程（1000 次以上），数据不易挥发（至少保存 10 年），使开发调试更为方便。随着社会的不断发展，人们使用各种家电设备、仪表以及工业生产中的数据采集与控制设备也在逐步走向智能化，所以充电器有它的巨大发展空间。而 51 单片机在实现手机电池充电器方面的应用就更具有意义了。

## 17.2　设计总体方案

实现智能化充电器需要从以下两个方面着手：

（1）充电实现　它包括两个部分：一是充电过程的控制；二是需要提供基本的充电电压。

（2）智能化实现　在充电器电路中引入单片机的控制。

所以本次设计综合了电子技术和单片机软硬件技术，选用了 89S51 单片机为核心，采用 MAX1898 作为充电芯片。系统主要有 89S51 单片机控制电路、充电电路、电压供电电路、

报警电路等构成。利用单片机的 T0 定时器，进行 C 语言编程。在安装好电池之后，输入直流电源，当充电器检测到电池时，定时器复位，从而进入预充状态。由 LED 来指示电池是否充电。当电池电压达到 2.5V，且电池温度都正常时，则进入快充状态。当电池充满电后，充电芯片 MAX1898 的第二个引脚 CHG 将会自动由低变高。CHG 信号的变化通过一个反相器送入到单片机的 INT0。单片机在接受到信号以后，通过软件程序将第 21 个引脚置于高电平，断掉给充电芯片提供的电压，并启动蜂鸣器，发出警报，提醒用户赶紧取下电池。

## 17.3　智能充电器实现原理及功能

### 17.3.1　智能充电器实现原理

本次设计的项目是设计一个智能充电器。主要所实现的系统功能就是能对任意一种锂电池进行自动预充、充电保护，并在充电完成后，实现自动断电，并产生蜂鸣器报警效果。本次设计是通过单片机来实现自动控制的功能。按照系统设计功能要求，确定系统 4 个模块组成：单片机智能控制模块、充电控制模块、供电电压模块、报警模块。主控制器采用的是 AT89S51，充电控制电路采用的是充电芯片 MAX1898。供电电压模块采用的是光耦模块 6N137 和 MOS 管 NDS332。报警模块采用的是蜂鸣器报警。其系统结构框图如图 17-1 所示。

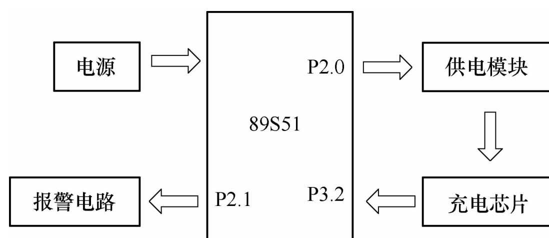



图 17-1　智能充电器的结构框图

对于主机控制供电电压模块的思路为当电池充满电后，充电芯片 MAX1898 能够自动检测到变化的信号，并使其引脚 CHG 由低变高，并将信号传给单片机，以实现及时关断充电电源。利用 PC/MCS-51 单片机联机开发系统编制和调试系统应用软件，并进行软硬件联调，最终实现以下几种功能：

**1. 预充**

在此期间检测到定时器，则进入预充。充电器以快充的 10% 给电池充电，使电池电压、温度恢复到正常状态。如果预充时间内电池电压达到 2.5V，且电池温度正常，则充电进入快充过程；如果超过预充时间后，电池电压仍低于 2.5V，则认为电池不可充电，充电器显示电池故障，LED 指示灯闪烁。

**2. 快充**

此时，充电器以恒定电流对电池充电。电池电压缓慢上升，一旦电池电压达到所设定的终止电压，恒流充电终止，充电电流快速递减，充电进入满充过程。

**3. 满充**

在满充过程中，充电电流逐渐衰减，直到充电速率降到设置值以下或满充时间超时，转入顶端截止充电。顶端截止充电时，充电器以极小的充电电流为电池补充能量。

**4. 断电**

当电池充满后，MAX1898 发送的脉冲电平由低变高，被单片机检测到后，引起中断，

并切断对 MAX1898 的供电，保证芯片和电池安全。

**5. 报警**

当单片机检测到充满状态的脉冲后，通过蜂鸣器报警，提醒用户及时取出电池。当充电出错时，MAX1898 会控制 LED 以 1.5Hz 左右的频率闪烁，以告知用户。

### 17.3.2 智能充电器功能分析

根据设计的任务要求，选用 89S51 单片机为核心，采用 MAX1898 作为充电芯片，组成智能充电器。系统主要由 89S51 单片机控制电路、充电电路、供电电压电路、报警电路构成。通过充电芯片 MAX1898 内部检测电池是否充满电，并在电池充满后自动发出信号。单片机在接收到不同的指令，输出不同的命令，以控制相应的报警电路，以及充电芯片的供电电压。

本设计中控制芯片采用的是 89S51 单片机，各个功能通过不同模块实现，主要有单片机最小系统、充电控制模块、供电电压模块、报警模块，具体实现方式如下：

供电电压模块：采用一个光耦模块 6N137 和模式管 NDS332 来控制对充电芯片 MAX1898 提供供电电压。当 6N137 的第三个引脚输入为低电平时，内部的发光二极管点亮，整个芯片工作，并且有一个较低的电平输出。再通过 NDS332 的放大作用，给充电芯片 MAX1898 提供足够的供电电压。当此引脚输入为高电平时，OUTPUT 输出为高，模式管 NDS332 不导通，就无法给充电芯片 MAX1898 足够的供电电压，即芯片 MAX1898 不工作。

充电控制模块：采用的是芯片 MAX1898 来实现充电过程的控制。MAX1898 的内部电路包括输入电流调节器、电压检测器、充电电流检测器、定时器、温度检测器和主控制器。当安装好电池后，该芯片能够自动检测到，并将其定时器复位以实现充电。也能够自动检测到电池充满后，使其引脚 CHG 由低变高，并将信号传给单片机，以实现及时关断充电电源。

报警电路：利用蜂鸣器来进行报警提醒用户。

## 17.4 智能充电器硬件电路设计

手机电池的使用寿命和单次使用时间以及充电过程密切相关。锂电池在生活中使用非常广泛，虽然有很多的优点，但它对于充电器的要求比较苛刻，需要保护电路。并且为了有效利用电池容量，需要将锂电池充电至最大电压，但过电压充电会造成电池的损坏。而在此次智能充电器的设计过程中主要的侧重点是保证充电器对于充电电压的精确控制，设计中元件的类型也是按照这个重点来选择的。

### 17.4.1 单片机最小系统设计

AT89S51 是一款低功耗、高性能 CMOS 8 位单片机，可反复擦写 1000 次的 Flash 只读程序存储器。其各部分情况介绍如下：

（1）中央处理器（CPU） 单片机的核心，完成运算和控制功能，MCS-51 的 CPU 能处理 8 位二进制数或代码。

（2）内部数据存储器（内部 RAM） 89S51 芯片中有 128 个 RAM 单元，用于存放可读写的数据，简称内部 RAM。

（3）内部程序存储器（内部 ROM） 89S51 芯片内部有 8KB 掩膜 ROM，用于存放程序

或表格，因此称为程序存储器，简称内部 ROM。

（4）定时/计数器　89S51 共有两个可编程的 16 位定时/计数器，以实现定时或计数为功能。

（5）并行 I/O 口　MCS-51 共有 4 个可编程的 8 位的 I/O 口（P0，P1，P2，P3），以实现数据的并行输入/输出。

（6）串行口　MCS-51 单片机有一个全双工的串行 I/O 口，以实现单片机和其他设备之间的串行数据传送。

（7）中断控制系统　89S51 单片机具有 5 个中断源，两个中断优先级的中断控制系统，以满足控制应用的需要。

时钟电路 MCS-51 芯片的内部有振荡和时钟电路，但石英晶体和微调电容需外接。时钟电路为单片机产生时钟脉冲序列。系统允许的晶振频率为 2～12MHz。

本设计中单片机的最小系统设计如图 17-2 所示。

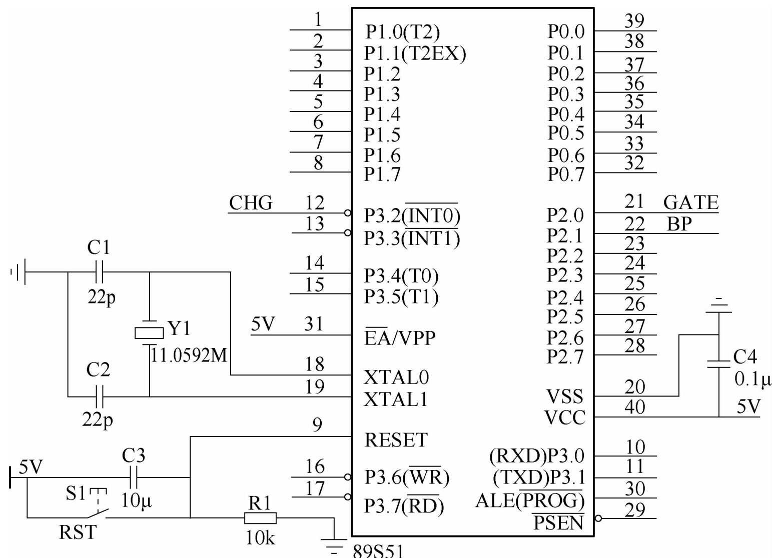



图 17-2　单片机最小系统设计

本设计中所需要用到的引脚说明：

①VCC：供电电压 5V，且通过一个 0.1μF 的电容接地，构成电源滤波电路；

②GND：接地端；

③RESET：接复位电路；

④XTAL0：接内部时钟工作电路的输入；

⑤XTAL1：接内部时钟工作电路的输出；

⑥$\overline{EA}$/VPP：当$\overline{EA}$保持低电平时，则在此期间访问外部程序存储器（0000H-FFFFH），不管是否有内部程序存储器。当$\overline{EA}$端保持高电平时，此期间访问内部程序存储器。本设计中，此引脚接高电平。

⑦P1.0：接一个跳线。人为设置一个外部中断，使得系统调试电路时更为直观。

⑧P2.0：与光耦合器 6N137 的第 3 个引脚相连接，用来控制是否给充电芯片 MAX1898

足够的电压，使其工作。当其输出为 0 时，光耦合器 6N137 的 OUTPUT 脚输出为低电平。当其输出为 1 时，光耦合器 6N137 的 OUTPUT 脚输出为高电平。

⑨P2.1：接报警电路，当其输出为 0 时，报警系统开启，蜂鸣器发出滴滴声。当其输出为 1 时，报警系统关闭。

⑩P3.2（INT0）：接充电芯片 MAX1898 的第 2 个引脚反相后的 CHG。

### 17.4.2 充电控制模块设计

因为市场上的手机电池大多数都为锂电池，所以本文中设计使用到的充电芯片为 MAX1898。这块芯片的内部电路包括输入电流调节器、电压检测器、充电电流检测器、定时器、温度检测器和主控制器。它的电池电压调节精度为 ±0.75%，能够提高电池性能并延长使用寿命。通过加上外部 PNP 或 PMOS 晶体管就可以对单节锂电池进行安全有效的充电。其最大的特点是具有精确的恒流/恒压充电。并在不使用电感的情况下，仍可以做到低功耗，且具有过电压保护和温度保护功能。其充电电流由用户设定，采用内部检流，无需外部检流电阻。其特点有：

①简单，安全的线性充电方式。

②使用低成本的 PNP 或 PMOS 调整元件。

③输入电压：4.5 ~ 12V。

④内置检流电阻。

⑤ ±0.75% 的电压精度。

⑥可编程充电电流。

⑦输入电源自动检测。

⑧LED 充电状态指示。

⑨可编程安全定时器。

⑩检流监视输出。

⑪可选/可调节自动重启。

MAX1898 的工作电路图如图 17-3 所示。

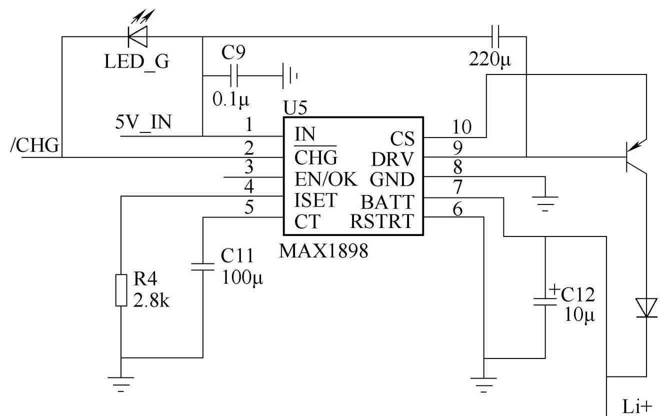



图 17-3 MAX1898 的工作电路

本设计中所需要用到的引脚说明：

①IN：电压输入，4.5 ~ 12V。

②CHG：输出以及充电状态指示脚，同时驱动 LED。当芯片检测到电池充满时，此引脚由低到高进行跳变。

③ISET：充电电流调节引脚，通过串接一个电阻到地来设置最大充电电流。

④CT：安全充电时间设置引脚。接一个时间电容来设置充电时间，电容为 100nF 时，几乎为 3 个小时，此引脚直接接地将禁用此功能。

⑤RSTRT：自动重新启动控制引脚。当此引脚直接接地时，如果电池电压掉至基准电压阈值以下 200mV，将会重新开始一轮充电周期。此引脚通过电阻接地时，可以降低它的电压阈值。此引脚悬空或者 CT 引脚接地（充电时间设置功能禁用）时，自动重新启动功能被禁用。

⑥BATT：电池传感输入脚，接单个锂电池的正极。此引脚需旁接一个大电解电容到地。

⑦GND：接地端。

⑧DRV：外部晶体管驱动器，接晶体管的基极。

⑨CS：电流传感输入，接晶体管的发射极。

MAX1898 可以通过外接限流型充电电源和 PNP 功率晶体管的线性调节来控制输入电流的大小。并且通过外接电容设定充电时间，通过 RSTRT 外接电阻来设置最大充电电流。

最大充电电流 $I_{FASTCHG}$ 和限流电阻 $R_{SET}$ 的关系式满足：

$$I_{FASTCHG} = \frac{1400}{R_{SET}}$$

式中，$I_{FASTCHG}$ 的单位是 A；$R_{SET}$ 的单位是 Ω。本设计中的 $R_{SET}$ 为 2.8k，所以最大充电电流 $I_{FASTCHG}$ 约为 500MA。

CT 引脚通过外接的电容可以设置充电时间 THG。这里的充电时间指的是快充时的最大充电时间，它和定时电容 $C_{CT}$ 的关系如下式所示：

$$C_{CT} = 34.33 \times T_{CHG}$$

式中，$T_{CHG}$ 的单位是 h，$C_{CT}$ 的单位是 nF。大多数情况下，快充时最大充电时间不超过 3h，因此常取 $C_{CT}$ 为 100nF。

其中，本设计中的模式管选用的是 NDS332，其最大电流可达 1A，以保证正常充电电流。

## 17.4.3　供电电压模块

本文所设计的用来提供充电电压的模块是由一个光耦合器 6N137 和一个模式管 NDS332 组成，之所以要接一个模式管 NDS332，是因为通过光耦合器 6N137 的内部消耗，它所输出的电流非常小，无法带动后面的负载。

6N137 光耦合器是一款单通道的高速光耦合器，内部有一个 850nm 波长 AlGaAs LED 和一个集成检测器组成，其检测器由一个光敏二极管、高增益线性运放及一个肖特基钳位的集电极开路的晶体管组成。

NDS332 是一个模式管。当 G 脚和 S 脚有电压差时，模式管开通，电流由 S 端流向 D 端。当 G 脚和 S 脚的有电压相等时，模式管截止，D 端电压为 0V。

提供电压的电路如图 17-4 所示。

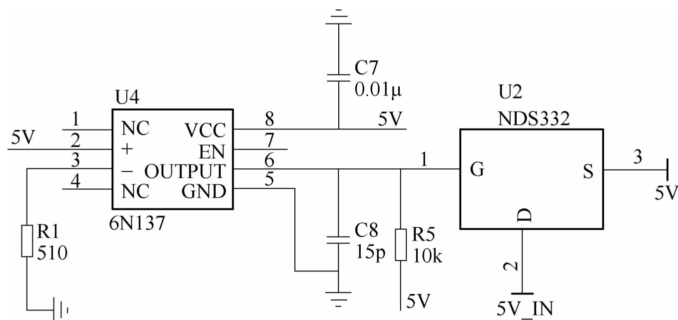



图 17-4 提供电压的电路图

光耦合器 6N137 的引脚说明：

①+：内部接的是发光二极管的正极，外部接的是 5V，用来驱动发光二极管。

②-：内部接的是发光二极管的负极，外部接一个 350Ω 的电阻，再连接到单片机的 P2.0 脚。当此引脚输入为低电平时，内部的发光二极管点亮，整个芯片工作，OUTPUT 脚输出为低电平。当此引脚输入为高电平时，内部的发光二极管不亮，即芯片不工作，OUTPUT 脚输出为高电平。

③GND：接地。

④VCC：供电电压 5V，且通过一个 0.01μF 的电容接地，用来保护芯片。

⑤OUTPUT：接模式管 NDS332 的 G 脚。并上拉一个 10k 的电阻到 5V。

模式管 NDS332 的引脚说明：

①G：作为输入端，接光耦合器 6N137 的 OUTPUT 脚。

②D：输出脚，为充电芯片 MAX1898 供电。

③S：接 5V 电压。

## 17.5 智能充电器软件实现

单片机的 C 语言采用 C51 编译器，由于 C51 产生的目标代码其运算速度高，所需存储空间小，符合 C 语言的 ANSI 标准，生成的代码遵循 Intel 目标文件格式，而且可与 A51 汇编语言或 PL/M51 语言目标代码混合使用。

应用 C51 编程具有以下优点：

（1）C51 管理内部寄存器和存储器的分配、编程时，无须考虑不同存储器的寻址和数据类型等细节问题。

（2）程序有若干函数组成，具有良好的模块化结构。

（3）有丰富的子程序库可直接引用，从而大大减少用户编程的工作量。

（4）C 语言和汇编语言可以交叉使用，汇编语言程序代码短，运行速度快，但复杂运算编程耗时。用汇编语言编写与硬件有关的部分程序，用 C 语言编写与硬件无关的运算部分程序，充分发挥两种语言的长处，提高开发效率。

### 17.5.1 单片机控制主程序设计

主程序的主要功能是初始化并等待外部中断的出现，单片机芯片在此设计中主要是对电

池起保护作用。当电池充满后，产生信号变化，使得单片机检测到外部中断，之后引起后面的一系列变化。

主流程如图 17-5 所示。

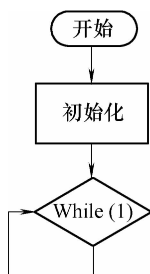



图 17-5　主程序流程图

```
void main( )
{
    TMOD =0x21;         //模式 1, T0 为 16 位定时/计数器
    ET0 =1;         //打开 T0 中断
    TR0 =1;         //T0 定时启动
    TR1 =1;
    SCON =0x50;         //REN =1 允许串行接受状态，串口工作模式 2
    PCON =0x80;         //波特率提高一倍
    TH1 =0xFD;      //baud * 2  / *  reload value 19200、数据位 8、停止位 1。效验位无
(11.0592)
    TL1 =0xF3;
    EA =1;         //CPU 总中断
    GATE =0;            //光耦正常输出电压
    BP =1;              //关闭蜂鸣器
    t_count =0;
    while(1);         / * 无限循环 * /
}
```

## 17.5.2　充电控制程序

充电器的充电过程主要由 MAX1898 控制。当 MAX1898 完成充电时，其$\overline{\text{CHG}}$引脚会产生由低到高的跳变，该跳变引起单片机的 INT0 中断。$\overline{\text{CHG}}$输出为高，存在两种情况：

①电池无充电输入。

②充电完毕。

其流程图如图 17-6 所示。

```
void timer0( ) interrupt 1 using 1
{
    TH0 =60536/256;         //5ms 定时
```

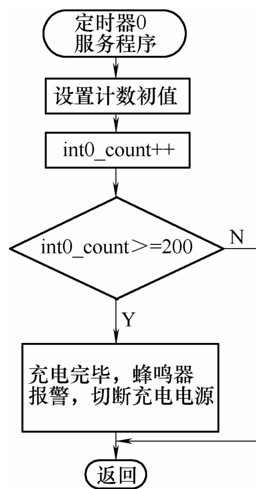



图17-6 充电控制流程图

```
    TL0 = 60536%256;
    if(t_count < = 15000)      //防止计数器累加溢出
        t_count ++ ;
    if(CHG)                    //如果 CHG 端口为高电平
        int0_count = 0;        //外部中断 0 的计数器
    else if(!CHG&&int0_count < = 10000)   //有外部中断,并且防止计数器溢出
        int0_count ++ ;
    if(int0_count > = 200)     //CHG 信号变低以后延时一秒钟
        {
        GATE = 1;              //关闭充电电源
        BP = 0;                //打开蜂鸣器报警
    }
    else //电未充满,或者模拟的 CHG 信号时间未到
    {
        GATE = 0;
        BP = 1;
    }
}
```

## 17.5.3 串口发送数据

为了方便观察程序运行到哪一步了，所以采用串口发送数据。串口的使用，能更快地检查出错误所在，以便更好地修改和纠正。

其主要程序如下：

```
void ComTX(char code * pStr)
{
```

```
    while( * pStr!='\0')        //地址上的内容不为空(传递时用地址来传送)
    {
        SBUF = * pStr;          //SUBF 接受/发送缓冲器(又叫串行通信特殊功能寄存器)
        while(!TI);             //等待数据传送(TI 发送中断标志)
        TI =0;                  //清除数据传送标志
        pStr ++ ;                   //下一个字符
    }
}
```

## 17.6　系统调试和结果分析

本文设计的智能充电器是采用 Protel99SE 环境进行电路图设计，在 Keil C51 集成环 uVision 中进行程序编写与调试，利用 ISP 和单片机烧写器下载程序到单片机内，最后硬件和软件联合调试，程序调试过程中用串口来观察程序运行到第几步，能更快地检查出错误所在，以便更好地修改和纠正。

### 17.6.1　电路原理图设计

Protel99SE 是桌面环境下第一个以设计管理和协作技术为核心的全方位印刷电路板设计系统，它集强大的设计能力，复杂工艺的可生产性和设计过程管理于一体，可完整实现电子产品从概念设计到生成物理生产数据的全过程，以及中间所有的分析，仿真和验证。

先进入 Protel99SE 主界面。执行 File/New 命令，进入一个新的项目设计，新建一个设计管理数据库文件。在 Document 下新建一个原理图文件，然后进入编辑窗口，进行原理图的设计与编辑。项目设计绘制好的电路原理图见附件清单。

### 17.6.2　程序调试

本设计中源程序的编写与调试是在 Keil C51 集成环境 uVision 中进行。Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，与汇编相比，C 语言在功能、结构性、可读性、可维护性上有明显的优势，因而易学易用。并且 Keil C51 软件提供丰富的库函数和功能强大的集成开发调试工具，全 Windows 界面。另外重要的一点，只要看一下编译后生成的汇编代码，就能体会到 Keil C51 生成的目标代码效率非常之高，多数语句生成的汇编代码很紧凑，容易理解。在开发大型软件时更能体现高级语言的优势。

首先建立一个新的工程，单击菜单 project，选择 new project，输入工程文件的名字，如保存到 keil 目录里，工程文件的名字为 Chongdiangi. c，单击保存。这时会弹出一个对话框，要求选择单片机的型号，可以根据使用的单片机来选择。单击菜单 File -> New，输入一个简单的程序，选择菜单 File -> SAVE，选择要保存的路径，如果是 c 程序文件，扩展名为 . c，单击 Target 1 前面的 + 号，展开里面的内容 source Group1，用右键单击 Source Group 1（注意用鼠标的右键，而不是左键），将弹出一个菜单，选择 Add Files to Guoup 'Source Group 1'，选择文件 Chongdiangi. c 后单击 Add，添加完毕此时再单击 Close 关闭该窗口。在 Source Group 1 里就有 Chongdiangi. c 文件。

程序在 Keil 软件下编译成功，如图 17-7 所示。


```
#include <reg51.h>
#define uchar unsigned char
#define uint unsigned int
#define ENTER   ComTX("\r\n")   //宏定义

sbit GATE = P2^0;
sbit BP = P2^1;
sbit CHG = P3^2;
sbit  MONI = P1^0;

uint t_count,int0_count;
//char Int0DelayEn;
/*************************************************************************
串口发送数据
*************************************************************************/
void ComTX(char code * pStr)
{
    while(*pStr != '\0')     // 地址上的内容不为空（传递时用地址来传送）
    {
        SBUF = *pStr;          //SBUF接受/发送缓冲器(又叫串行通信特殊功能寄
        while(!TI);            // 等待数据传送（TI发送中断标志）
        TI = 0;                // 清除数据传送标志
        pStr++;                   // 下一个字符
    }
}
/*************************************************************************
定时器0服务函数
*************************************************************************/
void timer0() interrupt 1 using 1

Build target 'Target 1'
assembling STARTUP.A51...
compiling chongdianqi.c...
linking...
Program Size: data=21.0 xdata=0 code=243
creating hex file from "quan"...
"quan" - 0 Error(s), 0 Warning(s).
```

图 17-7　程序编译成功界面

## 17.6.3　程序下载

在 Keil 软件下编译成功后，再用 ISP 和单片机开发板将正确的程序下载到单片机 AT89S51 中。其下载成功界面如图 17-8 所示。

## 17.6.4　结果分析

为了便于查询程序运行的步骤，在程序中加载串口发送数据。将串口的数据线接到单片机的 RXD 和 TXD 引脚，就能在 PC 上清楚地显示出来。

串口调试如图 17-9 所示，图中的“1”表示电路正在充电。

在充电芯片 MAX1898 的外围电路中，设置了其外接电容为 100nF，所以完整地充满一块电板需要大约 3h。为了在调试电路的时候便于观察，在单片机的 P1.0 口加了一个跳线。并在程序中设定一个模拟外部中断信号。当插槽拔出来的时候，CHG 将会变成低电平。当单片机检测到这一信号后，控制断电以及蜂鸣器报警。其模拟信号的串口调试如图 17-10 所示。

## 17.6.5　系统调试中所遇到问题

项目制作的实物如图 17-11 所示，在调试电路时遇到的问题如下：

1）最开始在设计供电电压模块的时候，直接采用光耦器件 6N137。由于器件内部消耗，

其 OUTPUT 端出来的电压只有 4.4V。而芯片 MAX1898 的供电电压为 4.5～12V，所以芯片 MAX1898 无法正常工作。于是就又在 6N137 之后接了一个 MOS 管 NDS332。将 6N137 的输出信号放大。




图 17-8　程序下载




图 17-9　串口调试




图 17-10 模拟信号串口调试

2）在调试单片机最小系统的时候，程序可以烧入，但芯片却始终不能工作。后来仔细研究了 89S51 单片机后，将其 EA 引脚接高电平就可以工作了。因为当$\overline{EA}$端保持高电平时，访问内部程序存储器。

3）对于蜂鸣器的连接，一开始让蜂鸣器的正极直接接电阻，然后拉高，将其负极接到单片机的 P2.1 脚。蜂鸣器一直都不会响。查看了蜂鸣器的用法之后，发现了问题所在。蜂鸣器的供电电压为 5V，电流为 20mA。而单片机的 I/O 口输出电流只有 10mA 左右。所以在蜂鸣器的前面加上一个 PNP 管，用来扩大信号。

4）在调试过程中，利用万用表测量各引脚导通情况，发现电路板中存在很多虚焊或者漏焊的现象。对于虚焊或者漏焊的都已重新焊接，测量无误。然后分开调试各个模块，最后再调试整个系统。

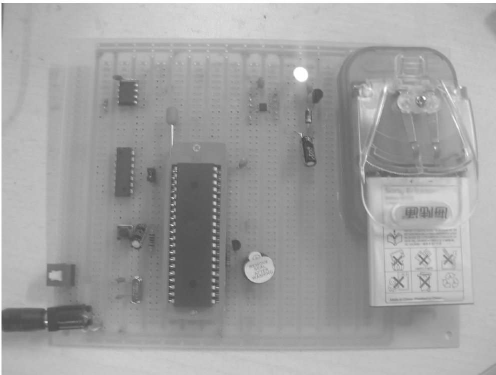


图 17-11 制作的智能充电器电路

## 17.7 总结

本系统设计并实现一个较为实用的智能充电器。采用单片机 89S51 作为核心控制，并用充电芯片 MAX1898 内部检测电池是否充满电，并在电池充满后自动发出信号。单片机接收到不同的指令，输出不同的命令，以控制相应的报警电路，以及断掉充电芯片的供电电压。避免电池因为过充而引起损伤。并且此次设计的智能充电器不但成本低，显示效果好，而且实用性、可靠性强。

## 附件：设计的电路原理图



图17-12 智能充电器电路原理图

# 第 18 章　无线遥控开关系统设计

## 18.1　项目背景及意义

### 18.1.1　项目背景

电器开关是经常使用、最常见、最普通的电子元件。所有的电器要运转起来都少不了它，可以说开关是电器和供电之间的桥梁。电的发明，特别是后来电器的出现，迫切需要一系列插件与之配合，便于使用电能。最早是没有电源开关之类的产品，为了电气连接，人们只能将电线绞拧在电源端子上。但是，随着电器的大量出现，如果每次都要绞拧电线，已太不适合需要了，而且有大量的非专业人员加入使用行列，这样就产生了很多的安全保障问题，弄不好就会造成严重的触电事故。这对电气连接提出了安全、方便、快捷的要求。在这种情况下，电器开关产品就应运而生了。

开关产品在不断地发展，最初只是满足简单取电需要。伴随人民生活水平的提高，安全意识的增强及对美观的需要，人们对插头插座也提出了更高的要求。目前，一大批智能型、遥控型、节能型的电源开关类产品正应运而生。

无线遥控开关的研究主要是关注于在家居智能化设计、工业安全行业上的应用，特别是工业上有很多危险环境威胁着人的生命安全，可以考虑采用无线遥控的方式来实现。

### 18.1.2　设计意义

无线技术已经成为工控领域核心技术，它将改变工业自动化的格局。目前，工业产品研发对于无线技术的需求不断增加，ARC 和 IMS 公司均进行了发展预测，全球自动化领域的无线通信应用市场预计在今后飞速增长。美国总统科技顾问委员会在“面向 21 世纪的联邦能源研究与发展规划”中指出：工业无线技术将使工业生产效率提高 10%，并使排放和污染降低 25%，可见无线技术不仅可以优化业务流程，还将为国家倡导的节能降耗做出突出的贡献。工业无线技术的潜力还需要我们进行深入的挖掘。

本项目设计的无线遥控开关具有性价比高，硬件施工维修方便，抗干扰性能好等特点，大大提高了生产效率和安全性。

## 18.2　方案论证

### 18.2.1　设计方案一

采用红外线发射、接收头组成的红外遥控器，如图 18-1 所示。红外遥控器（IR Rempte Control）是利用波长为 0.76 ~1.5μm 之间的近红外线来传送控制信号的遥控设备。红外遥

控器常用的载波频率为 38kHz，这是由发射端所使用的 455kHz 晶振来决定的。常用的红外遥控系统一般分发射和接收两部分。发射部分的主要元件为红外发光二极管。它实际上是一种特殊的发光二极管，由于其内部材料不同于普通发光二极管。因此在其两端施加一定电压时，发出的是红外线而不是可见光。目前大量使用的红外发光二极管发出的红外线波长为 940nm 左右，外形与普通发光二极管相同。只是颜色不同。接收部分的主要元件为红外接收二极管，一般有圆形和方形两种。在实际应用中要给红外接收二极管加反向偏压，才能正常工作，即红外接收二极管在电路中应用时是反向运用，这样才能获得较高的灵敏度。红外遥控的特点是不影响周边环境、不干扰其他电器设备。由于其无法穿透墙壁，故不同房间的家用电器可使用通用的遥控器而不会产生相互干扰；电路调试简单，只要按给定电路连接无误，一般不需任何调试即可投入工作；编解码容易，可进行多路控制。因此，现在红外遥控在家用电器、室内近距离（小于 10m）遥控中得到了广泛的应用。



图 18-1　红外线遥控器

### 18.2.2　设计方案二

采用无线电通信的无线电遥控器，如图 18-2 所示。无线遥控器（RF Remote Control）是利用无线电信号对远方的各种机构进行控制的遥控设备。常见的无线电发射接收模块一般分发射和接收两个部分。发射部分一般分为两种类型，即遥控器与发射模块，遥控器与遥控模块是对于使用方式来说的，遥控器可以当一个整机来独立使用，对外引出线有接线桩头；而遥控模块在电路中当一个元件来使用，根据其引脚定义进行应用，使用遥控模块的优势在于可以和应用电路天衣无缝的链接、体积小、价格低。接收部分一般来说也分成两种类型，即超外差与超再生接收方式，超再生解调电路也称超再生检波电路。它实际上是工作在间歇振荡状态下的再生检波电路。超外差式解调电路与超外差收音机相同，它是设置本机振荡电路产生振荡信号，与接收到的载频信号混频后，得到中频信号，经中频放大和检波，解调出数据信号。由于载频频率是固定的，所以其电路要比收音机简单些。超外差式的接收器稳定、灵敏度高、抗干扰能力也相对较好；超再生式的接收器体积小、价格便宜。



图 18-2　无线电遥控器

### 18.2.3　方案比较与选择

由于红外发光二极管的发射功率一般都较小（100mW 左右）。所以红外接收二极管接收到的信号比较微弱，因此就要增加高增益放大电路，最近几年大多都采用成品红外接收头。红外遥控和无线遥控是对不同的载波来说的，红外遥控器是用红外线来传送控制信号。它的特点是有方向性、不能有阻挡、距离一般不超过 7m、不受电磁干扰，电视机遥控器就是红外遥控器；无线电遥控器是用无线电波来传送控制信号的，它的特点是无方向性、可以不“面对面”控制、距离远（可达数十米，甚至数公里）、容易受电磁干扰。在需要远距离穿

透或者无方向性控制领域，如工业控制等，使用无线电遥控器较易解决，更符合本项目的设计要求，所以选择无线遥控方式。

## 18.3 无线遥控开关系统概述

### 18.3.1 工作原理

本设计分为遥控器和开关两个部分，为了节省成本及实现设计的功能要求，遥控器上设计了 4 个按钮来控制对应两路开关。遥控器上有 4 个按钮，其中 A 是开关 1 导通，C 是开关 1 断开，B 是开关 2 导通，D 是开关 2 断开，还有专门的发射模块组成。开关分为接收模块、专门的译码电路、单片机、数码管和继电器。A 按钮按下产生的高电平信号经过专门的编码芯片编码，将高电平变成一串载有数据的信号给发射模块，发射模块又将其变成载有数据的无线电波发送出去。开关上的接收模块把接收到的无线电波转换成载有数据的电信号，经专门的译码芯片译成高电信号，再经过单片机的信号处理，输出高电平，对应的高电平驱动继电器吸合使得开关导通同时数码管显示相应导通的开关键号，B 按钮按下时产生的高电平信号经过专门的编码芯片编码，将高电平变成一串载有数据的信号给发射模块，发射模块又将其变成载有数据的无线电波发送出去。开关上的接收模块把接收到的无线电波转换成载有数据的电信号，经专门的译码芯片译成高电信号，再经过单片机的信号处理，输出低电平使对应的驱动继电器释放，使得开关断开同时数码管显示控键标号，同理实现开关 2，进而实现本设计的要求。系统设计的电路结构图如图 18-3 所示。

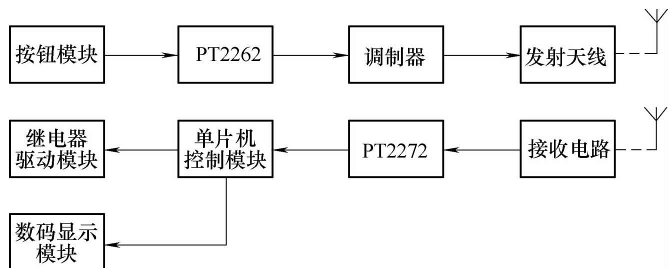



图 18-3 无线遥控开关系统结构框图

### 18.3.2 功能分析

通过综合应用所学电力电子技术和无线电技术实现一个较为实用的无线遥控开关，主要功能是通过遥控器来控制开关以及数码管显示。无线遥控开关是基于单片机对开关进行无线控制，在普通开关的结构上，采用专门的编译码芯片，以及相应的发射和接收模块等。遥控器的编码模块把按钮控制的电信号先编码，然后由发射模块转换成无线电波发送出去给接收模块。开关里的接收模块收到载有信息的无线电波后，把它放大，解码，得到原先的控制电信号，这个电信号再经过单片机，进行晶体管的功率放大，用来驱动继电器控制开关的闭合导通，以及数码管显示相应导电的开关键值，实现无线遥控以及显示功能。

# 18.4　无线遥控开关系统硬件设计

目前，市场上出现了很多无线数据收发模块，如 PTR2000、FB230 等。无线数据收发模块的性能优异，外围元器件少，设计、使用非常简单。无线收发模块一般在内部都集成了高频发射、高频接收、PLL 合成、FSK 调制/解调、参数放大和功率放大等功能。

在无线遥控领域，PT2262/2272 是目前最常用的芯片之一，项目设计主要使用这两个芯片。

## 18.4.1　发射模块

### 1. PT2262 编码芯片

PT2262 的引脚图及功能如图 18-4 所示。

无线遥控系统中编码和解码电路是关键，对于编码器 PT2262，A0 ~ A5 共 6 根线为地址线，而 A6 ~ A11 共 6 根线可以作为地址线，也可以作为数据线，这要取决于所配合使用的编码器。若解码器没有数据线，则 A6 ~ A11 作为地址线使用。这种情况下，A0 ~ A11 共 12 根地址线，每线都可以设置成“1”、“0”、“开路”三种状态之一，因此共有编码数 $3^{12}=531441$ 种。

图 18-4　PT2262 引脚图

编码芯片 PT2262 发出的编码信号由地址码、数据码、同步码组成一个完整的码字。解码芯片 PT2272 接收到信号后，其地址码经过两次比较核对后，VT 脚才输出高电平，于此同时相应的数据脚也输出高电平。PT2262 每次发射时至少发射 4 组字码，因为无线发射的特点是第一组字码非常容易受零电平干扰，往往会产生误码，所以 PT2272 只有在连续两次检测到相同的地址码加数据码，才会根据数据码中的“1”驱动相应的数据输出端为高电平，同时驱动 VT 端同步为高电平。当发射机没有按键按下时，PT2262 不接通电源，其第 17 脚为低电平，所以 315MHz 的高频发射电路不工作；当有按键按下时，PT2262 通电工作，其第 17 脚输出经调制的串行数据信号。当第 17 脚为高电平期间，315MHz 的高频发射电路起振并发射等幅高频信号；当第 17 脚为低电平期间，315MHz 的高频发射电路停止振荡，所以高频发射电路完全受控于 PT2262 的第 17 脚输出的数字信号，从而对高频电路完成幅度键控（ASK 调制），相当于调制度为 100% 的调幅。PT2262 各引脚功能描述见表 18-1。

**表 18-1　PT2262 引脚说明**

| 引　　脚 | 名　　称 | 说　　明 |
|---|---|---|
| 1 ~ 8、10 ~ 13 | A0 ~ A11 | 地址引脚，用于进行地址编码，可置为“0”，“1”，“f”（悬空） |
| 7 ~ 8、10 ~ 13 | D0 ~ D5 | 数据输入端，有一个为“1”即有编码发出，内部下拉 |
| 18 | VCC | 电源正端 |
| 9 | VSS | 电源负端 |
| 14 | TE | 编码启动端，用于多数据的编码发射，低电平有效 |
| 16 | OSC1 | 振荡电阻输入端，与 OSC2 所接电阻决定振荡频率 |
| 15 | OSC2 | 振荡电阻振荡器输出端 |
| 17 | Dout | 编码输出端（正常时为低电平） |

**2. 发射模块电路设计**

编码电路原理图如图 18-5 所示。

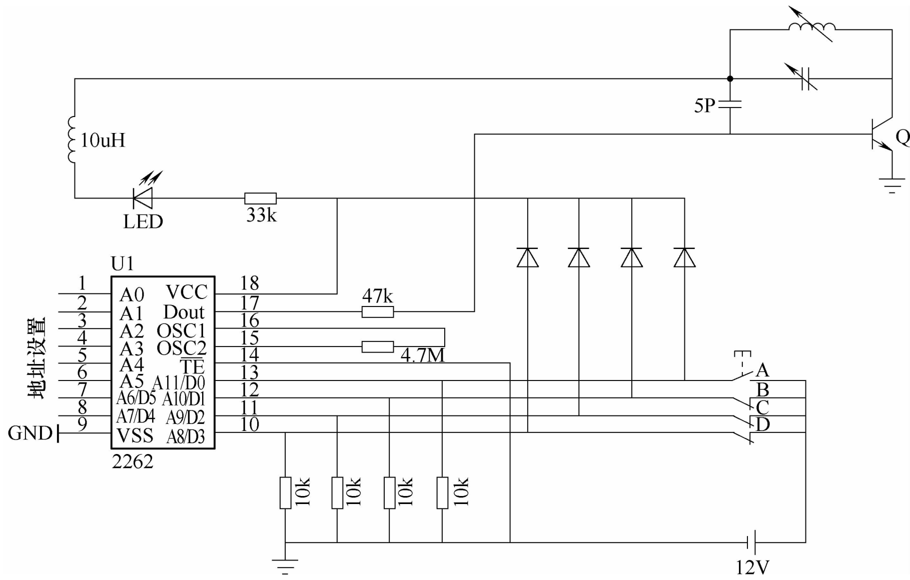



图 18-5 编码电路图

该部分电路主要由 315MHz 无线数据发射模块和编码集成在 PT2262 芯片中。该无线数据发射模块具有较宽的工作电压范围 3～12V，电压变化时发射频率基本不变，5 发射模块配套的接收模块无需任何调整就能稳定地接收。当发射电压为 3V 时，空旷地传输距离约 20～50m，发射功率较小；当电压 5V 时空旷地传输距离约 100～200m；当电压 9V 时空旷地传输距离约 300～500m；当发射电压为 12V 时，为最佳工作电压，具有较好的发射效果，发射电流约 60mA，空旷地传输距离为 700～800m，发射功率约 500mW。当电压大于 12V 时功耗增大，有效发射功率不再明显提高。发射模块采用 ASK 方式调制，以降低功耗。当数据信号停止时发射电流将为零，数据信号与 DF 发射模块输入端可以用电阻或者直接连接，而不能用电容耦合，否则 DF 发射模块将不能正常工作。数据电平应接近 DF 数据模块的实际工作电压，以获得较高的调制效果。

无按键操作时，晶体管 Q 截止，编码集成 IC1（PT2262）处于断电状态，无线数据发射模块没有发射信号。当按键 A、B、C、D 任何一个按下时，晶体管 Q 导通，编码集成 IC1 开始工作，它根据数据输入端 D0～D3 的电平进行编码，编码信号由地址码、数据码、同步码组成一个完整的码字。如果按钮一直按住，则发射模块连续发射无线信号。当编码集成 IC1（PT2262）第 17 脚为低电平期间，315MHz 的高频发射电路停止振荡，所以高频发射电路完全受控于 PT2262 的 17 脚输出的数字信号，从而对高频电路完成幅度键控（ASK 调制）相当于调制度为 100% 的幅度。发射模块电路结构图如图 18-6 所示。




图 18-6 发射模块电路结构图

## 18.4.2　无线遥控开关电路设计

该部分电路主要由 315MHz 无线数据接收模块、电源电路、单片机控制电路和继电器驱动电路构成。

**1. PT2272 解码芯片**

PT2272 的引脚功能　PT2272 编码芯片有不同的后缀，表示不同的功能，有 L4/M4/L6/M6 之分，其中 L 表示锁存输出，数据只要成功接收就能一直保持对应的电平状态，直到下次遥控数据发生变化时才改变；M 表示非锁存输出，数据引脚输出的电平是瞬时的而且与发射端是否发射相对应，可用于类似点动控制；后缀的 6 和 4 表示有几路并行的控制通道，当采用 4 路并行数据时（PT2272-M4），对应的地址编码应该是 8 位，当采用 6 路的并行数据时（PT2272-M6），对应的地址编码应该是 6 位。PT2272 的数据输出有“暂存”和“锁存”两种，“暂存”是当输入端信号消失时，PT2272 对应的数据位输出变成低电平；“锁存”是当输入端信号消失时，PT2272 的数据位输出保持原有状态，直到接收到地址码相同的新输入。PT2272 引脚图如图 18-7 所示，各引脚功能描述见表 18-2。

图 18-7　PT2272 引脚图

**表 18-2　PT2272 引脚说明**

| 引　脚 | 名　称 | 说　明 |
|---|---|---|
| 1～8、10～13 | A0～A11 | 地址引脚，用于进行地址编码，可置为“0”，“1”，“f”（悬空）必须与 2262 一致，否则不解码 |
| 7～8、10～13 | D0～D5 | 地址或数据引脚，当为数据引脚时，只有地址码与 2262 一致，数据引脚才能输出与 2262 数据端对应的高电平，否则输出为低电平，锁存型只有在接收到下一数据才能转换 |
| 18 | VCC | 电源正端 |
| 9 | VSS | 电源负端 |
| 14 | DIN | 数据信号输入端，来自接收模块输出端 |
| 16 | OSC1 | 振荡电阻输入端，与 OSC2 所接电阻决定振荡频率 |
| 15 | OSC2 | 振荡电阻振荡器输出端 |
| 17 | VT | 解码有效确认输出端（常低）解码有效变成高电平（瞬态） |

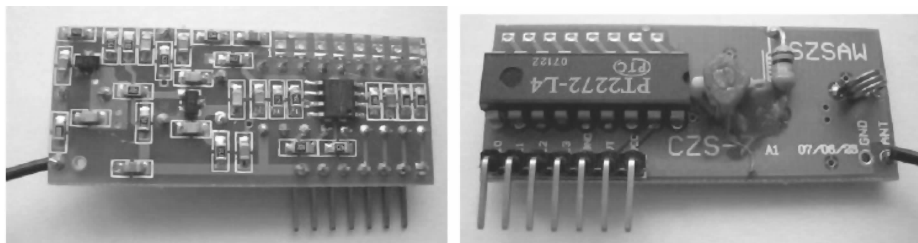
**2. PT2272 接收模块**

频率 315MHz，振荡电阻 200k，编解码芯片均兼容 PT2262/2272 系列。接收模块从工作方式分可以分成超外差接收板和超再生接收板。超再生式接收机具有电路简单、性能适中、成本低廉的优点，所以被本设计采用。接收电路模块如图 18-8 所示。

接收模块采用 SMD 贴片工艺制作生产，为超再生接收方式，它内含放大整形及编码电路，使用极为方便。

（1）由于天线输入端具有选频电路，所以可以不依赖 1/4 波长天线进行选频，通信距离较近时可以剪短甚至去掉外接天线。

（2）接收电路自身辐射极小，加上电路模板背面网状接地铜箔的屏蔽作用，可以减少自身振荡的泄露和外界干扰信号的侵入。




接收板正面图　　　　接收板反面图

图 18-8　接收电路模块

（3）接收机采用高进度带骨架的铜心电感将频率调整到 315MHz 后封固，这与采用可调电容调整接收频率的电路相比，温度、湿度稳定性及抗机械振动性能都有极大改善。可调电容调整精度较低，只有 3/4 圈的调整范围，而可调电感可以做到多圈调整。可调电容调整完毕后无法封固，因为无论导体还是绝缘体，各种介质的靠近或侵入都会使电容的容量发生变化，进而影响接收频率。另外未经封固的可调电容在受到振动时定片和动片之间发生位移；温度变化时热胀冷缩会使定片和动片间距离改变；湿度变化因介质变化改变容量；长期工作在潮湿环境中还会因定片和动片的氧化改变容量，这些都会严重影响接收频率的稳定性，而采用可调电感就可解决这些问题，因为电感可以在调整完毕后进行封固，绝缘体封固剂不会使电感量发生变化，而且由于采用贴片工艺，所以即使强烈震动也不必担心接收频点漂移，接收电路的接收带宽约 500kHz，产品出厂时已经将中心频率调整在 315kHz，接收电路的接收带宽约 500kHz，产品出厂时已经将中心频率调整在 315MHz，接收芯片上的微调电感约有 5MHz 频率的可调范围，使用时不要轻易变动，以免影响性能。

模块中解码电路使用的为 PT2272 解码芯片。PT2272 工作电压为 5V，其中 A0 ~ A11 为地址引脚，用于进行地址编码，可置为“0”，“1”或“f”（悬空），但必须与 PT2262 一致，否则不解码。D0 ~ D3 为数据引脚，当地址码与 PT2262 一致时，数据引脚输出与 PT2262 数据端对应的高电平，锁存器只有在接收到下一数据时才能转换。DIN 为数据信号输入端，来自接收模块输出端。OSC1、OSC2 分别为振荡电阻的输入和输出端，外界电阻决定振荡频率。VT 为解码有效确认端，高电平有效。

平时，无线接收模块没有接收到空间的 315MHz 信号时，解码集成 PT2272-L4 输出端 D0 ~ D3 均为低电平。当无线接收模块接收到空间的 315MHz 信号时，经放大、变频、滤波等处理后输出控制信号，送到第 14 脚进行解码。只有 PT2272 的地址端的电平状态与发射部分的 PT2262 的地址端一致时，对应的数据端才有高电平输出。本设计的地址为 FFFFFF，即全部悬空。设计的电路图如图 18-9 所示。

**3. 单片机最小系统**

本设计中控制电路主要使用的是 AT89S51 单片机最小系统。主要包括：单片机芯片、复位电路、时钟电路。如图 18-10 为单片机 AT89S51 的最小系统电路图，由 5V 电源供电。

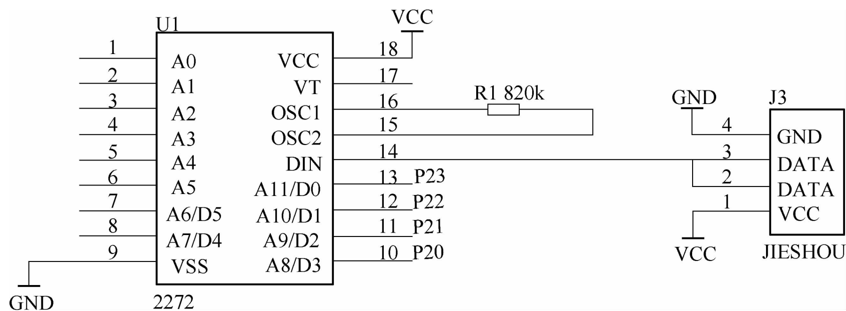



图 18-9　接收模块电路图

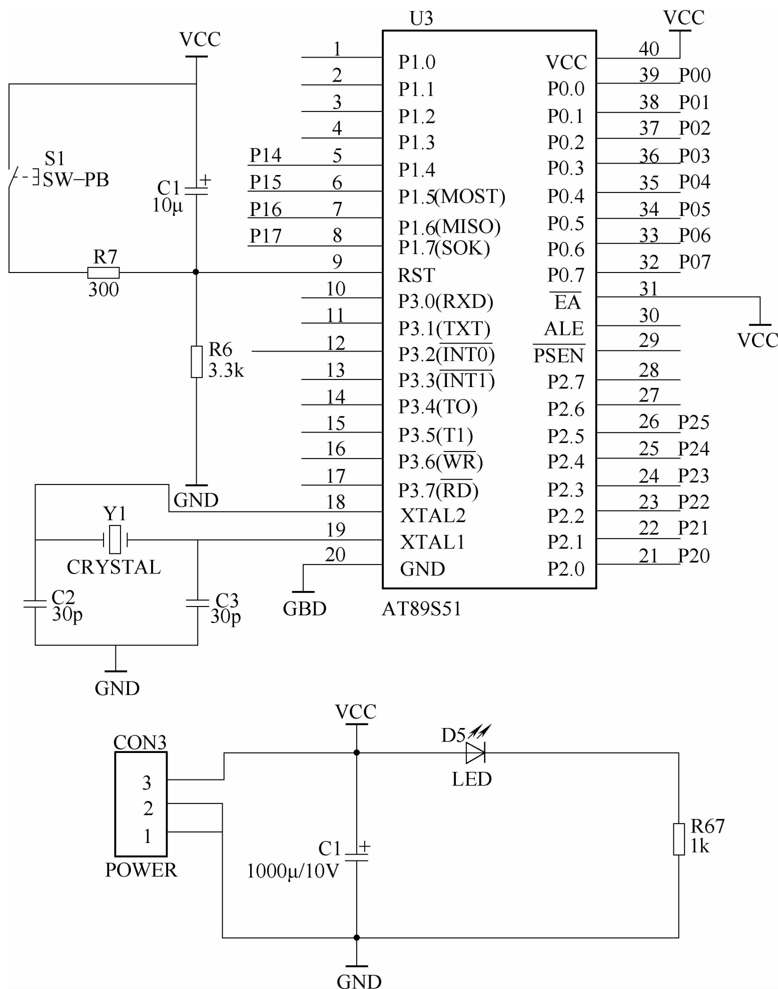



图 18-10　AT89S51 单片机最小系统电路图

（1）复位电路

单片机的复位是为了把电路初始化到一个确定的状态，一般来说，单片机复位电路作用是把一个状态机初始化到空状态，单片机内部复位时，对寄存器以及存储器装入厂商预设的一个值。复位电路如图 18-11 所示。

单片机复位电路原理是在单片机的复位引脚 RST 上外接电阻和电容，实现上电复位。当复位电平持续两个机器周期以上时复位有效。复位电平的持续时间必须大于单片机的两个机器周期。具体数值可以由 RC 电路计算出时间常数。复位电路由按键复位和上电复位两部分组成。51 系列单片机高电平复位，通常在复位引脚 RST 上连接一个电容一端接至 VCC，再连接 1 个电阻到地端，由此形成一个 RC 充放电回路，保证单片机在上电时 RST 引脚上有足够时间的高电平进行复位，随后回归到低电平进入正常工作状态，这个电阻电容的典型值为 10k 和 10μF。

按键复位就是在复位电容上并联一个开关，当开关按下时电容被放电、RST 也被拉到高电平，而且由于电容的充电，会保持一段时间的高电平来使单片机复位。

（2）振荡电路

单片机系统中晶振作用非常大，结合单片机内部电路产生单片机所需的时钟频率，单片机晶振提供的时钟频率越高，其运行速度就越快，单片机一切指令的执行都是建立在单片机晶振提供的时钟频率之上。振荡电路如图 18-12 所示。

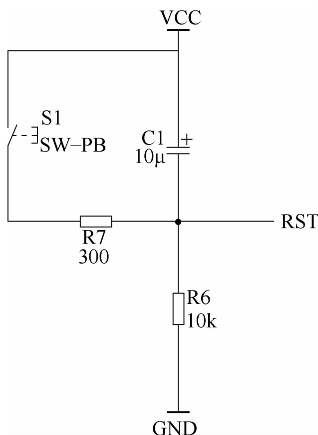


图 18-11　复位电路

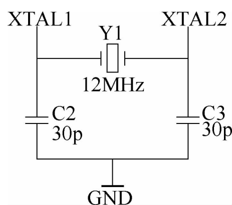


图 18-12　振荡电路图

单片机晶振的作用是为系统提供基本的时钟信号。通常一个系统共用一个晶振，便于各部分保持同步。有些通信系统的基频和射频使用不同的晶振，通过电子调整频率的方法保持同步。

（3）数码显示电路

本设计采用数码管显示对应的按键值，数显电路由数码管和单片机及其晶振电路和复位电路组成。选用的数码管是共阳数码管，数码管的段码端分别与单片机 P0 口相连，控制数码管段码显示，数码管的位选通由 4 个 PNP 晶体管控制，分别接到单片机的 P1. 4、P1. 5、P1. 6、P1. 7 引脚上，程序中通过控制 P1. 4 ~ P1. 7 引脚的输出电平控制数码管的显示与关闭。如 P1. 4 输出低电平时，晶体管导通，5V 电源加到第一个数码管的 COM 端，那么第一个数码管就会显示相应的数字，显示的数字由单片机 P0. 0 ~ P0. 7 输出段码决定，当 P1. 4 输出高电平时，晶体管截止，数码管就不显示，从而实现数码管位选控制。同理 P1. 5 输出低电平时，则第二个数码管显示。数码管显示电路如图 18-13 所示。

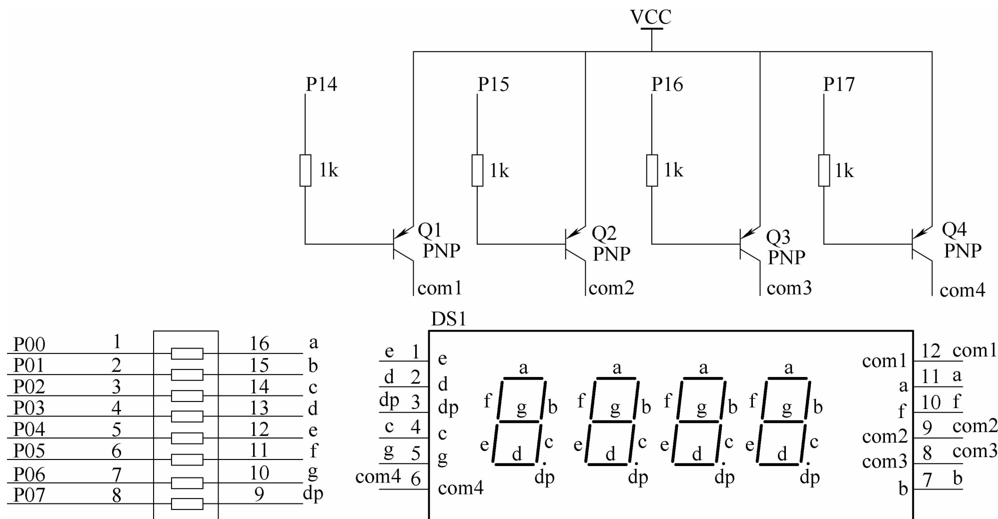



图 18-13　数码管显示电路

（4）继电器驱动电路

自动控制设备都存在一个电子电路（弱电）与电器电路（强电）的互相连接问题，一方便要使电子电路的控制信号能够控制电气电路的执行元件（如电动机、电磁铁和电灯等），另一方面又要为电子线路的电气电路提供良好的电隔离，以保护电子电路的人身安全。继电器便能起到桥梁作用。本设计通过单片机来控制继电器吸合、释放，实现用较小电流去控制较大电流的过程。

继电器的工作原理就是当电磁铁的绕组中有电流通过时，衔铁被电磁铁吸引，因而就改变了触电的状态。

本设计中是利用单片机通过一个 NPN 型晶体管，把晶体管作为电子开关来驱动继电器，继电器的开和关完全由晶体管的基极电平进行控制。当晶体管基极为高电平，NPN 型晶体管导通，继电器通电吸合；反之低电平的话，NPN 型晶体管截止，这时继电器不工作。继电器驱动电路如图 18-14 所示。

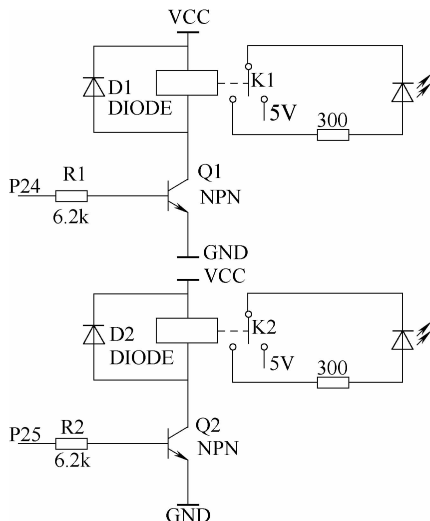



图 18-14　继电器驱动电路

当遥控器上按下 A 按钮，发射模块 PT2262 对信号进行编译并通过天线将信号发射出来，接收模块 PT2272-L4 解译模块对信号进行解码，通过 P2.0 脚发出高电平到单片机，由于采用的 PT2272-L4 解码模块是锁存的，所以会连续不断地把高电平发送到单片机，单片机又把高电平发送到对应的 P2.4 脚上，对应的 NPN 型晶体管导通，继电器通电吸合，LED 亮。当遥控器上按 B 按钮，发射模块 PT2262 对信号进行编译并通过天线将信号发射出来，接收模块 PT2272-L4 连续不断地把高电平发送到单片机，此时单片机对信号进行识别，同时输出低电平信号

到 P2.4，对应的 NPN 型晶体管截止，此时继电器断开，LED 熄灭。

## 18.5 无线遥控开关软件设计

### 18.5.1 开关无线接收程序设计

当遥控器按下按钮，此时通过 PT2262 编码后将信号通过天线传送出来。在无线遥控开关接收时，程序中首先是将 P2 口初始化，接收模块将接收到的信号传送到 PT2272-L4 中进行译码，并且将信号电平通过 P2.0 ~ P2.3 传送给单片机，单片机根据 P2.0 ~ P2.3 的低四位来决定 P2.4 ~ P2.5 的输出。P2.0 接收到高电平时，经过单片机处理相应的 P2.4 输出高电平，晶体管导通继电器闭合，LED 亮。当 P2.1 接收到高电平时，经过单片机处理，相应的 P2.4 输出低电平，晶体管截止，继电器断开，LED 灭。同理 P2.2 ~ P2.3 也是如此。程序流程图如图 18-15 所示。

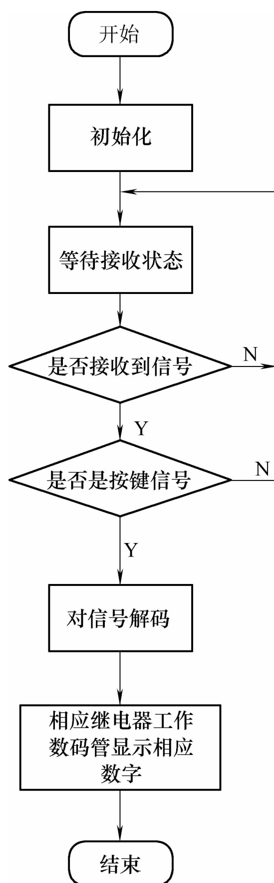



图 18-15 主程序流程图

开关接收数据关键代码如下：

```
void main(void)
{
    char datavalue;
```

```
    P1 = 0x00;                    //端口初始化
    P0 = 0x00;                    //端口初始化
    P2 = 0xff;                    //置输入状态
    while(1)
    {
  dat = (P2&0x0f);
  if(dat == 0x02)                 //先按 A 显示 1 通道 灯亮 通道开
{
  datavalue = 0x01;               //数码显示
  P24 = 1;
}
  if(dat == 0x01)                 //再按 C 键 灯灭通道关
{
  P24 = 0;
}
  if(dat == 0x04)                 //按 B 键 显示 2 通道 开
{
  datavalue = 0x02;               //数码显示
  P25 = 1;                        //灯亮
  }
  if(dat == 0x08)                 //再按 D 键 2 通道关
  {
  P25 = 0;
  }
  display(datavalue);             //将读到的数显示
  NOP();
  }
}
```

## 18.5.2　数码显示程序设计

当遥控器按下按钮，此时 PT2262 编码后将信号通过天线传送出来。无线遥控开关接收时，程序中首先是将 P2 口初始化，接收模块将接收到的信号传送到 PT2272-L4 中进行译码，并且将信号电平通过 P2.0 ~ P2.3 传送给单片机，单片机根据 P2.0 ~ P2.3 的低四位来决定 P0.0 ~ P0.7 的输出。P2.0 接收到高电平时，经过单片机处理，数码管显示 0001。当 P2.1 接收到高电平时，经过单片机处理，数码管显示 0000。同理 P2.2、P2.3 原理分析相同。

数码显示部分关键代码如下：

```
void display(int k)
{
    P1 = 0xfe;                      //位选
```

```
    P0 = tab[k/1000];              //显示千位数字
    delay();                    //延时
    P1 =0xfd;                   //位选
    P0 = tab[k%1000/100];          //显示百位数字
    delay();                    //延时
    P1 =0xfb;                   //位选
    P0 = tab[k%100/10];         //显示十位数字
    delay();                    //延时
    P1 =0xf7;                   //位选
    P0 = tab[k%10];                //显示个位数字
    delay();                    //延时
    P1 =0xff;                   //位选
}
```

## 18.6 系统调试

### 18.6.1 程序编译

Keil 是由美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。

打开 Keil 软件，编译调试程序，在程序编译无错的情况下，生成 . hex 文件。项目程序编译成功界面如图 18-16 所示。

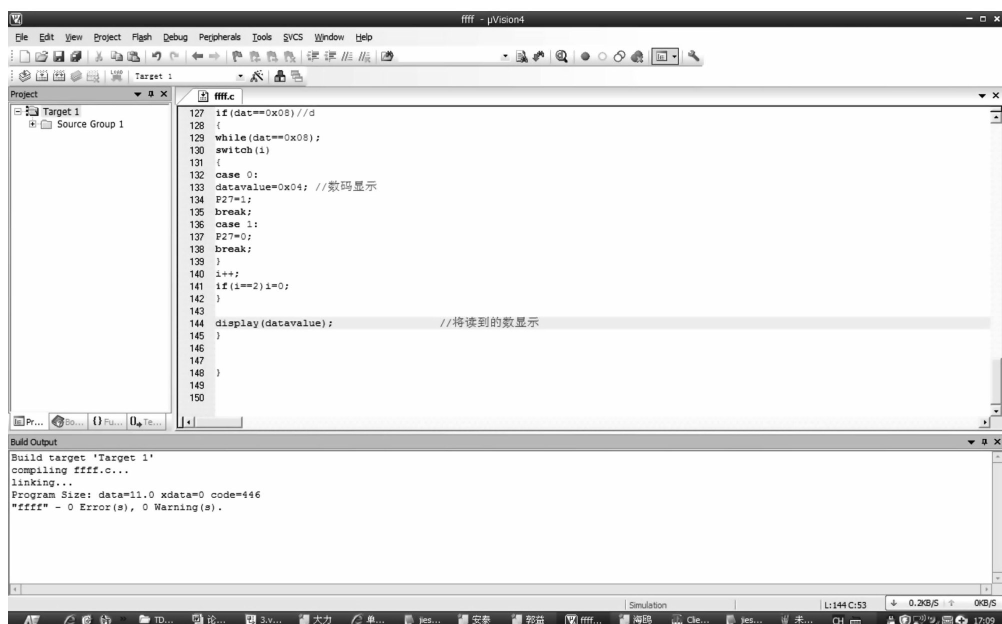



图 18-16 项目程序编译成功界面

## 18.6.2　程序下载

在 Keil 软件下编译成功后，再用 ISP 和单片机开发板（或者程序烧写器）将编译成功的程序下载到单片机 AT89S51 中。其下载成功的界面如图 18-17 所示。




图 18-17　程序烧写进程

将下载好程序的单片机装入插槽中，对照电路图和实际线路检查连线是否正确，包括错接、少接、多接等。用万用表电阻挡检查焊接是否良好；元器件引脚之间有无短路，连接处有无接触不良，二极管、晶体管、集成电路和电解电容的极性是否正确；电源供电包括极性、信号源连线是否正确；电源端对地是否存在短路等，做好前期检查工作为下一步调试做准备。

## 18.6.3　调试出现的问题

系统根据电路板设计的基本流程，先设计电路原理图，产生网络表，再设计印制电路板 PCB，报表输出。将 PT2262 和 PT2272 两芯片的地址码一致，PT2272 数据引脚才能输出与 PT2262 数据端对应的高电平，否则输出为低电平，锁存器只有在接收到下一数据才能转换。为了方便调试，开关部分调试过程采用 LED 模拟仿真。

经过软件编译和系统仿真调试进行实物电路制作，大部分问题出现在硬件的电路板子上。若电路经过检查，确定无误后，断开信号源，把经过准确测量的电源接入电路，用万用

表电压挡检测电源电压，观察有无异常现象；如冒烟、异常气味、元器件发烫、电压短路等，如发现异常就马上切断电源，排除故障。如没有发现异常情况，分别测量各关键点的直流电压，如静态工作点、数字电路各输入端和输出端的高、低电平值及逻辑关系、放大电路输入/输出端直流电压等是否正常工作。若有，就排除故障，使电路最终工作在正常的工作状态下。而对于放大电路必须要用示波器观察。最后在电路的输入端输入所需的信号源，即按下遥控上的按钮，并分别按下 A、B、C、D 按键逐级检测各有关的波形、参数和性能指标是否满足设计要求。

## 18.7　总结

本设计相对于普通开关的亮点主要有无线遥控发射与接收功能，数字显示功能。而且具有扩展性，安全性高，操作方便可靠。通过本项目学习，方便读者能够更好地掌握并利用单片机控制技术，无线遥控技术，以及相关开发工具的应用，采用 C 语言编程控制技术解决单片机软件控制问题，通过动手操作让自己的动手能力和实践能力在实践过程中得到一定的提高。

## 附件：设计的电路原理图

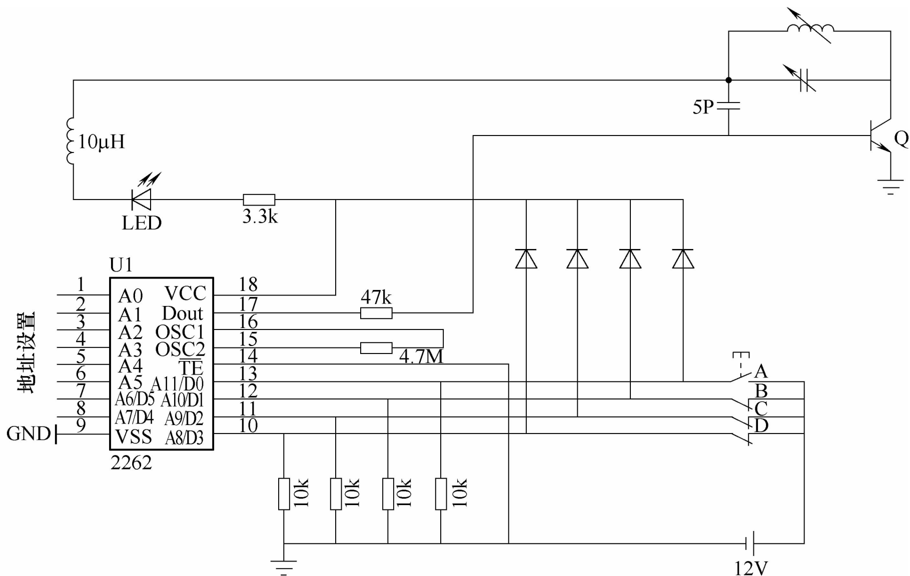



图 18-18　发射模块电路原理图

图18-19　接收模块电路原理图

# 第 19 章　融合物联感知与 GSM 的果园环境监测系统设计

## 19.1　项目说明

### 19.1.1　研究背景

物联网作为计算机互联网在实物世界的延伸，被世界多个主要国家看作今后主要的经济增长点，也是现在业界和学术界研究的热点之一。与此同时，具有较强数据通信能力的第三代移动通信网络已经在世界范围内得到了迅速普及，应用业务支持更为广泛的第四代移动通信系统也开始投入应用推广阶段，能够实现无缝覆盖、接入方便的移动通信网络为物联网的实际应用提供了坚实的物质基础。

由于物联网信息节点的广泛性和移动性，决定了各种无线通信技术将是物联网的主要联网技术。同时，随着第三代移动通信的不断发展，普及现代移动通信网络的数据通信功能日益强大，已经开始应用的 4G 通信网络支持的业务范围更加广泛。因此现代移动通信网络为物联网的实现提供了很好的物质基础，移动通信系统必将在物联网中的组网过程中得到广泛应用。

移动通信系统一般由移动终端、传输网络和网络管理维护等部分组成，因此移动通信在物联网的应用主要包括以下几个方面：

（1）移动通信终端在物联网中的应用　移动终端作为信息接入的终端设备，可以随网络信息节点移动并实现信息节点和网络之间随时、随地通信。对比移动通信终端和物联网节点信息感知终端的功能和工作方式可知，移动通信终端完全可以作为物联网信息节点终端的通信部件使用。

（2）移动通信传输网络在物联网中的应用　移动通信系统的传输网络主要实现各移动节点的相互连接和信息的远程传输，而物联网中的信息传输网络也是完成相类似的功能。因此，完全可以将现有移动通信系统的信息传输网络作为物联网的信息传输网络使用，即可将物联网承载在现有的移动通信网络之上。

（3）移动通信网络管理平台在物联网中的应用　移动通信网络的网络管理维护平台主要用来实现对网络设备、性能、用户及业务的管理和维护以保证网络系统的可靠运行。为了保证信息的安全、可靠传输物联网同样需要相应的管理维护平台以完成物联网相关的管理维护功能。因此可以将移动通信网络管理维护的相关思想、架构应用到物联网的网络管理和维护中。

### 19.1.2　研究现状

中国农业科技不但在研究水平方面与发达国家存在着明显的差距，而且在技术推广应用

方面存在更大的差距。据中国农科院估测，中国农业科技研究开发的总体水平与国际先进水平相差 10 ~ 15 年，而农业生产的技术水平与发达国家的差距则在 15 ~ 20 年以上。农业机械、农机具有的研究开发与制造技术，农机具有的性能、质量技术水平，只相当于 20 世纪 80 年代世界先进水平，落后近 20 年。信息技术在我国农业领域的应用虽起步较晚，但发展很快，我国无论是在信息传播高速公路的硬件建设方面，还是在农业信息平台和资源建设方面都取得了长足的进展，令世人刮目相看。主要表现在以下方面：数据库建设初具规模；涉农网站发展迅速；电信电视网快速发展；农村信息服务深入基层；农业信息技术研发成果显著。

目前，欧美国家农业信息技术的发展已进入产业化阶段，以美国、德国和日本为代表的发达国家在完成了农业工业化和农业机械化后已经进入农业信息化时代，形成了从农业信息的采集、加工处理到发布比较健全、完善的农业信息技术体系。

随着农业信息技术，特别是数字农业技术地迅速发展，美国、欧洲、日本等发达国家建立了大量的农业经济和农业资源数据库系统，如联合国粮农组织存取数据库（AGRIS）、国际食物信息数据库（IFIS）、美国农业部农业联机存取数据库（AGRICOLA）、国际农业生物中心数据库（CABI）等，并广泛用于农业生产管理的方案制定以及产业结构调整的优化决策。在北美和澳大利亚，大力发展以遥感、地理信息系统、全球定位系统（俗称 3S）与智能决策支持系统及智能化农业机械系统相结合的精确农业，为农庄进行精确施肥、喷药、灌溉和测产等。欧洲共同体将信息与通讯技术（ICT）在农业上的应用列为重要发展规划，将农业专家系统、决策支持系统、地理信息系统、数据仓库系统等紧密结合，以数据挖掘（DW）、联机分析（OLAP）以及多种智能优化技术为手段，应用于农产品价格预测、农业生产的规划、管理、效益分析。韩国、埃及、印度、巴西、菲律宾等许多发展中国家均重视信息技术在农业上的应用。

### 19. 1. 3　研究内容

针对传统果苗生长环境信息获取科学度低、时效性差等不足，将物联感知与 GSM 技术应用到果苗生长环境监测系统中。通过对果苗生长所需的空气和土壤温湿度信息进行采集，利用 GSM TC35i 模块以短消息的方式实现数据远程传输，实时将采集的信息发送到果农手机端。系统需要完成的功能如下：

（1）研究以 GSM 模块 TC35i 为基础的短信收发控制功能。

（2）研究无线数据远程监测系统，利用传感技术以及移动通信网络，对果苗生长所需环境的空气温湿度以及土壤湿度信息进行实时监测，并通过现场显示和短信的方式对用户实时通报。

（3）研究以 89C51 单片机为主控制器的传感器采集电路，可以将采集的信息远程发送到果农手机。

## 19. 2　果园环境监测系统方案设计

### 19. 2. 1　系统结构原理

果园环境信息远程监测系统主要是对果苗生长参数（空气温湿度和土壤温湿度）进行

远程监测。图 19-1 为果园环境信息远程监测系统原理图，整个系统由底层的远程监测采集感知端、中间层移动 GSM 网络和上层移动客户端三部分构成。采集感知端通过对果苗生长参数进行数据采集，利用控制器对采集数据进行处理，利用串口与 GSM 模块进行通信，通过 SMS AT 指令将数据信息依靠移动网络发送到用户终端，用户端对短信进行 PDU 解码，使用户随时掌握果苗生长参数信息。

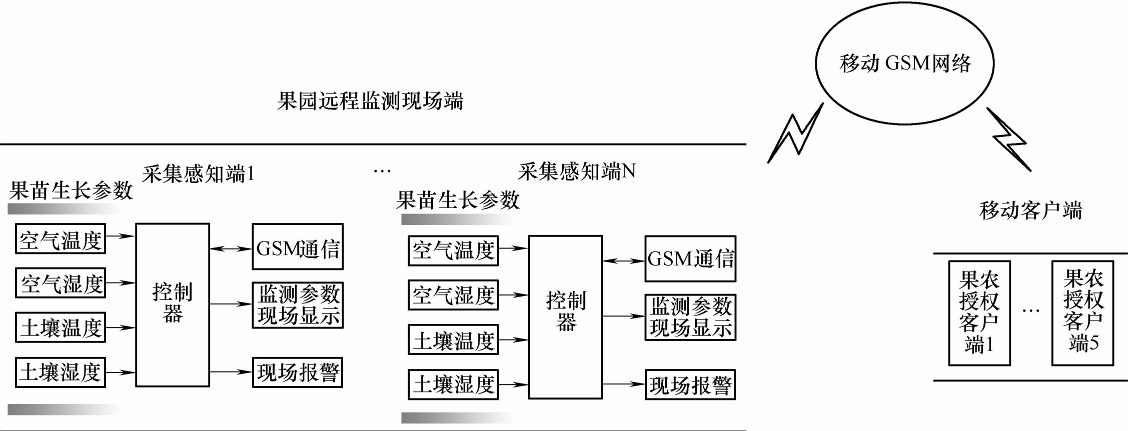



图 19-1　系统整体结构原理图

图 19-2 为现场端数据采集电路框图，其中果苗生长所需要空气温度和湿度数据被传感器 DHT11 感知，土壤温度和湿度数据被 SLHT5-1 感知，通过单片机 P1 口和 P3 口读入感知数据并计算空气和土壤的温湿度值。其中串行时钟输入 SCK 用于单片机与传感器之间的通信同步，DATA 用于数据的读取。单片机 P2 口将感知的空气温湿度和土壤温湿度信息通过液晶 LCD 在现场端实时显示，依据果苗正常生长信息，系统预先设定监测数据合理的门限值范围，若传感器感知的参数信息超过或者低于门限值，则单片机通过 P3 口控制报警电路，发出警情提示。

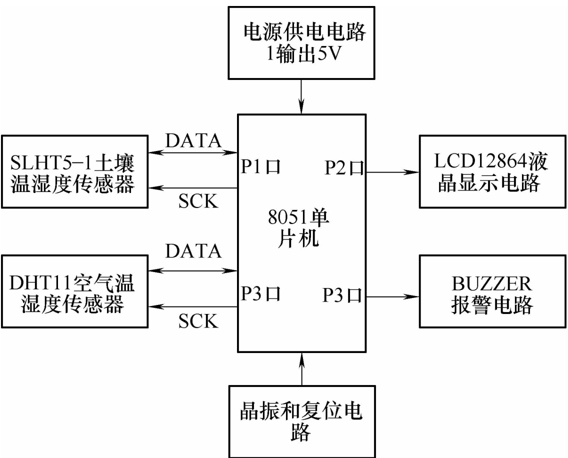



图 19-2　现场端数据采集电路框图

单片机将采集的数据通过串行通信口 P3.0 和 P3.1，经电平转换电路与 GSM 模块 TC35i 相连，控制 TC35i 对数据进行收发处理。

其中，GSM 模块负责将采集的信息通过 GSM 网络发送到远程用户端。区别于单片机的供电电压 5V，GSM 模块的供电电压为 4.2V。GSM 模块有 AT 命令集接口，支持文本和 PDU 模式的短消息，通过 40 脚 ZIF 连接器，实现指令、数据及控制信号的双向传输。图 19-3 为单片机与 GSM 模块接口电路图，通过 ZIF 连接器和 GSM 模块分别外接 SIM 卡存储电路、模块启动电路和同步指示电路。

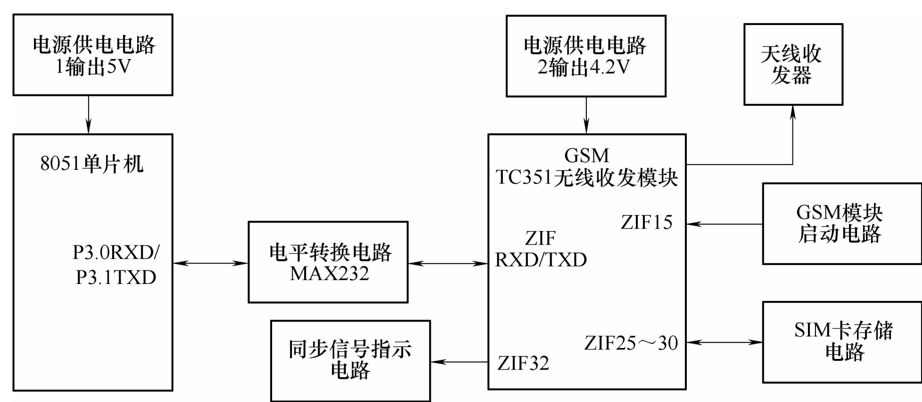



图 19-3　单片机与 GSM 模块接口电路图

SIM 卡卡座中的引脚分别与 TC35i 模块上的 ZIF 连接器 25 ~ 30 脚相连，引脚将 SIM 卡中的数据读出供 TC35i 模块使用。启动电路需给 ZIF 连接器 15 脚加时长至少为 100ms 的低电平信号，且该信号下降沿时间小于 1ms，系统启动后 ZIF 连接器 15 脚的信号应保持高电平。同步指示电路为 ZIF 的 32 脚，当 LED 熄灭时，模块处于关闭或睡眠状态；当 LED 为 600 ms 亮/600ms 灭时，表明未插入 SIM 卡或正在搜索网络；当 LED 为 75ms 亮/75s 灭时，表明模块已登录进网络，处于待机状态；闪烁，表明数据传输中。

### 19. 2. 2　系统功能分析

系统工作时，用户通过移动 GSM 网络，融合物联感知技术远程监测果苗生长参数信息。

（1）果园环境信息现场采集感知端　现场端通过传感器 DHT11 感知果苗生长环境的空气温湿度，传感器 SLHT5-1 感知土壤温湿度，并对感知的信息实时采集，处理器采用 8051 单片机对采集的信号进行处理，一方面将处理的数据实时通过液晶 LCD 在现场端显示，如果任何一项监测指标超过或低于设定门限值，现场监测电路发生报警，鸣响 30s。另一方面感知的数据通过 GSM 通信模块利用移动网络发送到用户端。

（2）移动客户端　用户发送控制指令到采集感知端，感知端在 10 ~ 30s 自动向用户端回传短消息显示当前监测数据。如果监测指标不在监测范围内现场报警，现场监测电路会自动发送报警短信到授权的用户手机端，为了保证监测的信息能够准确的发送给用户，预先设定 5 个号码，逐次向已授权的 5 个号码发送短信，用户根据相应短信内容实时对果苗生长环境进行远程监测。

## 19. 3　果园环境远程监测系统电路设计

### 19. 3. 1　单片机最小系统

单片机最小系统由单片机、晶振电路和复位电路组成。晶振的作用是为系统提供基本的时钟信号。通常一个系统共用一个晶振，便于各部分保持同步。

时钟振荡电路为单片机提供时钟脉冲，通过 XTAL1 ~ XTAL2 引脚与电容和晶体振荡器

构成振荡电路。时钟振荡电路内置反馈电阻，为了在停止模式中降低功耗，将反馈电阻从振荡电路分开，单片机最小系统电路图如图 19-4 所示。

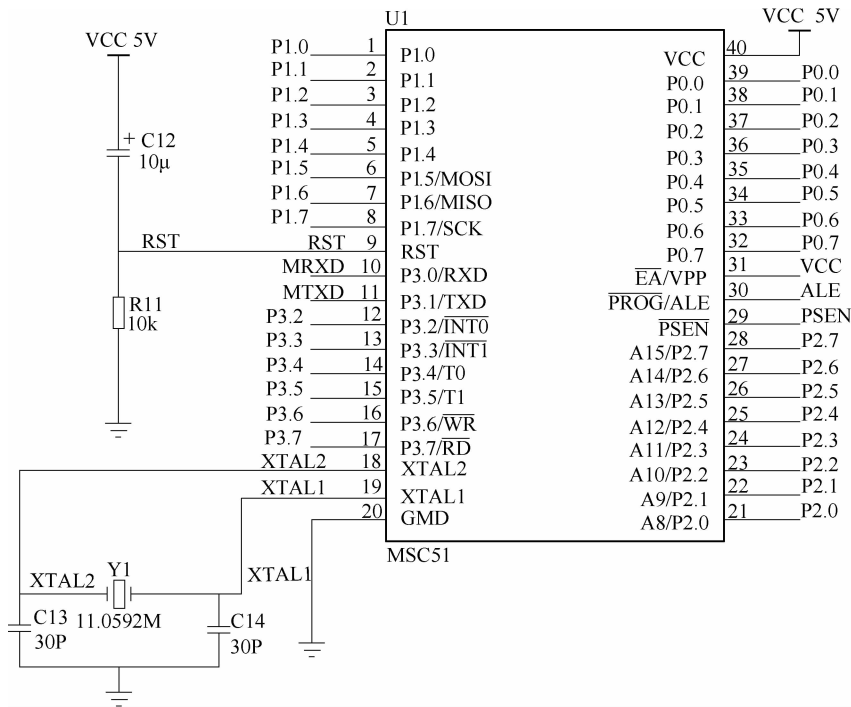



图 19-4　单片机最小系统电路

## 19.3.2　现场端采集电路

现场端采集信号主要感测空气的温度和湿度以及土壤的温度和湿度，针对空气温度和湿度监测，选择数字温湿度传感器 DHT11。

DHT11 是一款含有已校准数字信号输出的温湿度复合传感器，应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与稳定性。传感器由一个电阻式感湿元件和一个 NTC 测温元件组成，DHT11 与单片机连接采用单线制串行接口，系统集成简单快捷，信号传输距离 20m 以上，具有 4 针单排引脚封装。DHT11 实物图如图 19-5 所示。

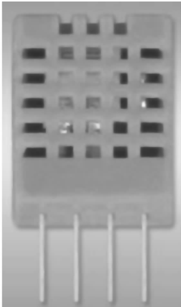


图 19-5　DHT11 实物图

DHT11 引脚说明：

1 脚 VDD：供电电源 3 ~5. 5V DC。

2 脚 DATA：串行数据，单总线。

3 脚 NC：空脚，悬空。

4 脚 GND：接地，电源负极。

DHT11 上电后，要等待 1s 以越过不稳定状态，在此期间无需发送任何指令，电源和地端之间可以增加一个 100nF 的电容，用于电源去耦滤波。DATA 用于单片机与 DHT11 之间的通信和同步，采用单总线数据格式，一次通信时间 4ms 左右，数据分小数部分和整数部分，完成的数据传输为 40 位，高位先出。

数据格式：8 位温度整数数据 +8 位湿度小数数据 +8 位温度整数数据 +8 位温度小数数

据 +8 位校验和。

数据传输正确时校验和数据等于 8 位温度整数数据 +8 位湿度小数数据 +8 位温度整数数据 +8 位温度小数数据所得结果的末 8 位。单片机发送一次开始信号后，DHT11 从低功耗模式转换到高速模式，等待单片机开始信号结束后，DHT11 发送响应信号，送出 40 位数据，并触发一次信号采集，用户可以选择读取部分数据，从模式下 DHT11 接收到开始信号触发一次温湿度采集，如果没有接收到主机发送开始信号，DHT11 不会主动进行温湿度采集，采集数据后转换到低速模式。通信过程如图 19-6 所示。

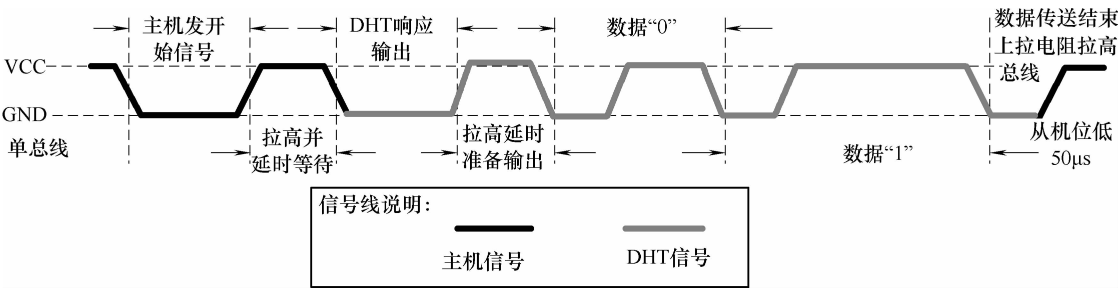



图 19-6　DHT11 通信过程

总线空闲状态为高电平，主机把总线拉低等待 DHT11 响应，拉低时间必须大于 18ms，保证 DHT11 检测到起始信号。DHT11 接收到主机的开始信号后，等待主机开始信号结束，然后发送 80μs 低电平响应信号。主机发送开始信号结束后延时等待 20 ~ 40μs 后，读取 DHT11 的响应信号。

若总线为低电平，DHT11 发送响应信号，把总线拉高 80μs，准备发送数据，每一位数据以 50μs 低电平时隙开始，如果读取响应信号为高，则 DHT11 没有响应，检查线路是否连接正确；若最后 1 位数据传输完成后，DHT11 拉低总线 50μs，则总线上拉电阻拉高进入空闲状态。图 19-8 给出 DHT11 电路连接原理。

SLHT5-1 土壤型温湿度探头实物图如图 19-7 所示。采用原装进口温湿度传感器为核心部件，可直接连接单片机使用。具有非常高的一致性，可完全互换，温度测量精度可达到 ±4.5% RH。传感器采用接插式分立设计，4 芯接插件，便于更换与维护，铜烧结网的防护加强了探头的耐温、耐压、耐损能力，适合于农业温室大棚、花卉、苗圃、草坪等需要检测土壤温湿度环境的场合使用。

SLHT5-1 传感器特点：

①全量程标定，两线数字输出。

②湿度测量范围：0 ~ 100% RH。

③温度测量范围：−40 ~ 123.8℃。

④湿度测量精度：±4.5% RH。

⑤温度测量精度：±0.5℃。

⑥响应时间：8s（tau63%）。

⑦低功耗 80μW（12 位测量，1 次/s）。

⑧可完全被水浸没。

使用方法：可将探头直接埋入土壤中。



图 19-7　SLHT5-1 土壤型温湿度探头实物图

接线方式：VCC-红色（电源正极）、GND-黑色（电源负极）、DATA-蓝色（数据）、SCK-时序（黄色）。

图19-8给出SLHT5-1电路连接原理图。

图19-8所示为现场端数据采集电路原理图，其中果苗生长所需要空气温度和湿度数据被传感器DHT11感知，土壤温度和湿度数据被SLHT5-1感知，通过单片机P1口和P3口读入感知数据并计算空气和土壤的温湿度值。其中串行时钟输入SCK用于单片机与传感器之间的通信同步，DATA用于数据的读取。单片机P2口将感知的空气温湿度和土壤温湿度信息通过液晶LCD在现场端实时显示，依据果苗正常生长信息，系统预先设定监测数据合理的门限值范围，若传感器感知的参数信息超过或者低于门限值，则单片机通过P3口控制报警电路，发出警情提示。

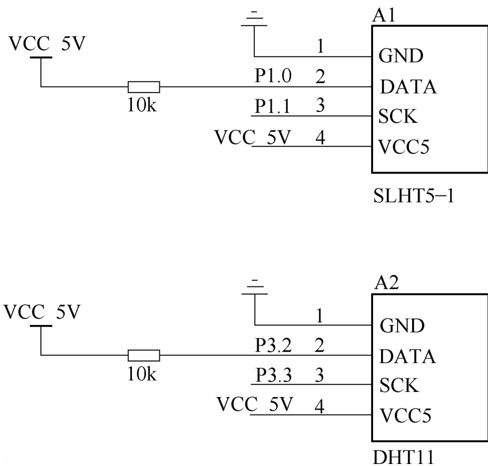



图19-8 现场端数据采集电路原理图

### 19.3.3 GSM TC35i外围电路设计

**1. SIM卡电路**

SIM卡卡座中的引脚分别与TC35i模块上的ZIF连接器25～30脚相连，引脚将SIM卡中的数据读出供TC35i模块使用。SIM卡接口电路如图19-9所示。

**2. IGT启动电路**

启动电路由开漏极晶体管和上电复位电路组成。模块上电10ms后（电池电压须大于3V），为使之正常工作，必须在15脚（$\overline{\text{IGT}}$）加时长至少为100ms的低电平信号，且该信号下降沿时间小于1ms。启动后，15脚的信号应保持高电平。如图19-10所示，为启动电路产生的信号，从中可以得到10ms的延时和100ms的低电平。

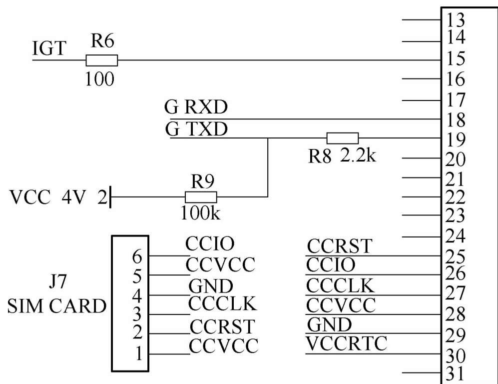



图19-9 SIM卡接口电路

**3. 同步电路**

同步指示电路为ZIF的32脚，当LED熄灭时，模块处于关闭或睡眠状态；当LED为600ms亮/600ms灭时，表明未插入SIM卡或正在搜索网络；当LED为75ms亮/75s灭时，表明模块已登录进网络，处于待机状态；闪烁，表明数据传输中。同步电路图如图19-11所示。

**4. 液晶显示电路**

液晶12864的RS，RW，EN分别与单片机的P2.0，P2.1，P2.2相连，实现使能信号传输；液晶12864的数据端DB0～DB7与单片机的P0.0～P0.7相连实现数据传输，液晶左半屏与右半屏的使能信号CS1，CS2端口分别连接单片机的P2.3，P2.4，起到显示作用。液晶RST复位引脚接单片机的P2.5脚，对液晶复位时，将P2.5脚置低电平，用来显示空气的温

湿度或者土壤的温湿度的值。液晶接口电路连接电路图如图 19-12 所示。

图 19-10　IGT 启动电路

图 19-11　同步电路图

图 19-12　液晶显示接口电路图

**5. 警情信号提醒电路**

蜂鸣器与一个晶体管相连，晶体管主要是做驱动作用。因为单片机的 IO 口驱动能力不够让蜂鸣器发出声音，所以通过晶体管来放大驱动电流，从而可以让蜂鸣器发出声音，单片机 P3.4 引脚输出高电平，晶体管导通，集电极电流通过蜂鸣器发出声音，当输出低电平时，晶体管截止，没有电流流过蜂鸣器，所以就不会发出声音。警情提醒电路如图 19-13 所示。

**6. 电平转换电路**（max232）

考虑单片机输出 TTL 电平，一般 TTL 电平特点：

输出高电平 >2.4V，输出低电平 <0.4V。在室温下，一般输出高电平是 3.5V，输出低电平是 0.2V。最小输入高电平和低电平：输入高电平≥2.0V，输入低电平≤0.8V，噪声容限是 0.4V。

而 TC35i 输出 CMOS 电平，一般 CMOS 电平特点：

1 逻辑电平电压接近于电源电压，0 逻辑电平接近于 0V，而且具有很宽的噪声容限。

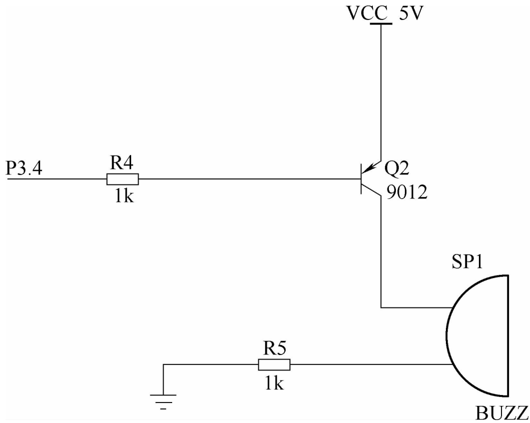



图 19-13　警情提醒电路图

所以考虑到 TTL 和 COMS 的高低电平的值不一样，所以互相连接时需要电平的转换，转换电路选择 max232 进行电平转换，电路如图 19-14 所示。

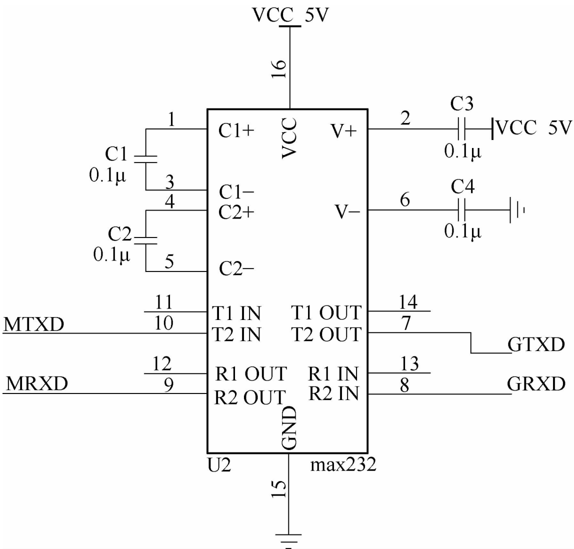



图 19-14　max232 电平转换电路

TC35i 的 18 脚 RXD、19 脚 TXD 为 TTL 的串口通信脚，通过图 19-14 的电平转换电路与单片机进行通信。TC35i 提供一个 8 线，不平衡的，异步的串行接口，与 ITU-T V.24 协议的 DCE 发出的信号一致。TC35i 专为 DCE 设计，基于 DCE-DTE 连接协议，它可与用户应用通过如下信号进行交流（DCE）：DTE 端口/TXD 发送数据到 DCE 模块的/TXD0 信号端；DTE

端口/RXD 接收从模块 DCE 传输的/RXD0 信号。具体电路连接如图 19-15 所示。

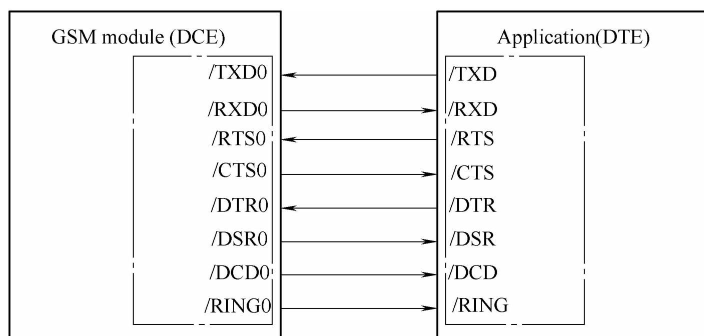



图 19-15　DCE-DTE 连接

## 19.4　果园环境监测系统现场感知端软件实现

### 19.4.1　主程序设计

现场感知端软件实现的重点在于单片机通过串口向 GSM TC35i 模块发送 SMS AT 指令实现短消息的发送和接收，用户可以不受通信距离的限制实时掌握果苗生长信息。主程序设计的软件流程图如图 19-16 所示。关键代码分析如下：

```
void main(void)
{
delayms(100); //上电,等待稳定
init_com();     //串口初始化
SendString("AT\r\n");
lcd_init();   //LCD 初始化
GSM_Init(); //GSM 初始化
RocoverySysValue();//上电数据恢复
Beep();         //beep
ReadTel();      //读取授权号码
Menu2();        //界面 2
s_connectionreset();//SHTxx 初始化
Timer_init();//Timer 初始化
    while(1)
    {
      Read_SHTxx();//读土壤温湿度
      Read_DHT11();//读空气温湿度
      CheckKeypress();//按键检查
      CheckOver();//检查是否超出标准值
```

```
        FillData(menu_index);//数据更新
        RecvMsgHandle();//短信接收处理
    }
}
```

## 19.4.2　现场端数据信息发送程序设计

对于 SIM 卡中的短信息，可以显示其类型、发送者号码、信息正文和发送时间，当发送短信息时应该先设置并检查串口是否打开和 TC35i 模块是否正确连接，还要判断目标手机和短信息中心号码位数是否正确，之后再发送 AT 命令，发送短信息程序流程图如图 19-17 所示。

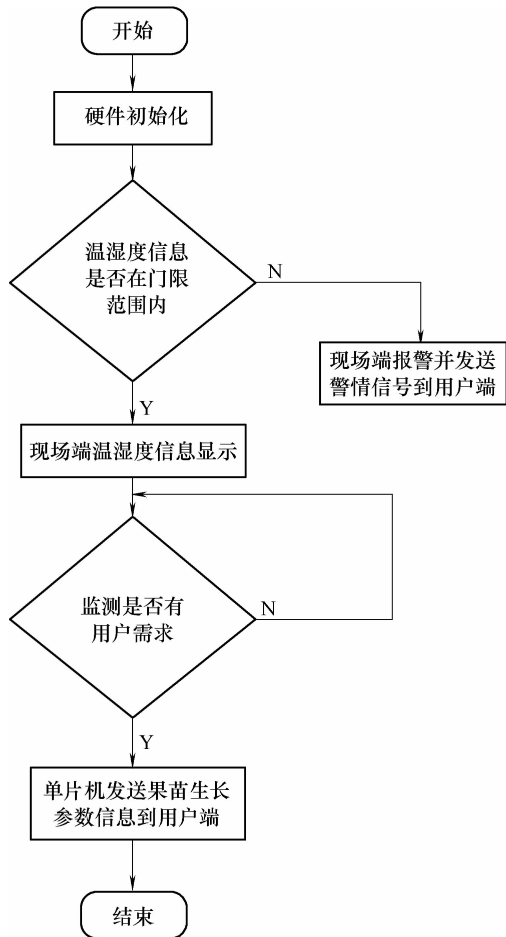



图 19-16　主程序流程图

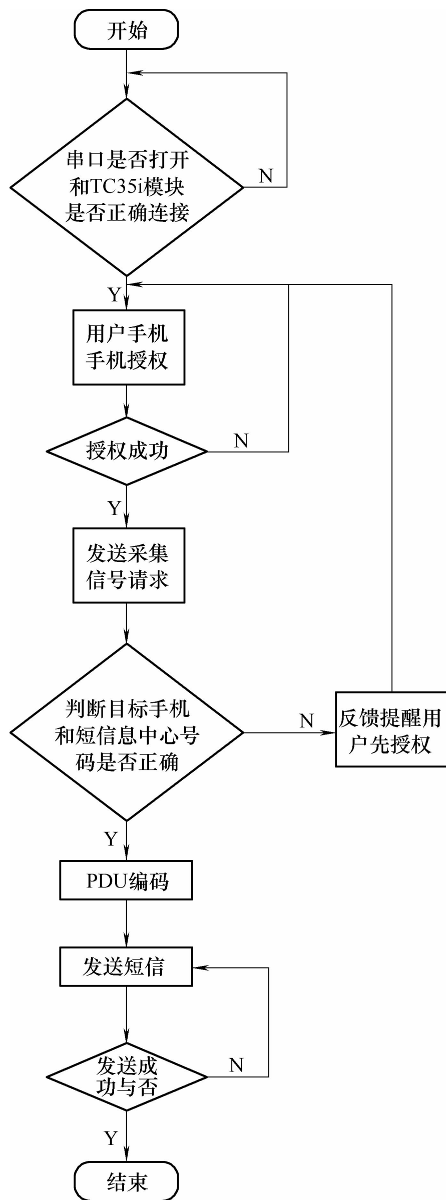



图 19-17　短信发送的程序流程图

```
char GSM_Init(void)
{
    TC35_IGT = 1;
    delayms2(200);
    TC35_IGT = 0;
    delayms2(200);
    TC35_IGT = 1;
    SendString("AT + CNMI = 2,1,0,0\r\n"); //通过字符串发送函数显示短信发送的
消息
    delayms2(100);
    ClearBuffer(buff,100);
    return 0;
}
```

单片机通过 GSM TC35i 模块发送 AT 指令和 PDU 数据包实现短消息的发送，设定短消息中心的指令格式：AT + CSCA = “ + 8613800571500”，向用户发送短信格式：AT + CSCA = + 8615904091469，短信内容：“air moisture：60%”，则 GSM 模块发出 PDU 数据包编码如下：0891683108501705F511000B815109041964F90008A72000610069007200200006D006F00690073007400750072006500720065FF1A003600300025，利用短信业务的 PDU 模式将 SMS 地址、中心电话号码、用户号码和空气湿度信息进行 Unicode 编码，压缩为 PDU 数据包进行发送。字符串发送代码如下：

```
void SendString(unsigned char *dat)
{
    while(*dat! ='\0')              //判断字符串是否发送完毕
  {
    SendChar(*dat);             //发送单个字符
  dat++;                        //字符地址加 1,指向下一个字符
  }
}
```

### 19.4.3　现场端数据信息接收程序设计

若 TC35i 接收到用户反馈信息，则通过“AT + CMGR”命令，读取短信息的 PDU 数据包，依据解析的代码执行相应的 AT 指令。接收短信息程序流程图如图 19-18 所示。

现场感知端通过设定 AT + CMGR 指令读取用户发送的信息，并对接收到的信息解码，从短消息的 PDU 数据包中获得接收所用用户号码。

```
void ReadMsg(char a)  //读短信
{
    SendString("AT + CMGR = ");
    SendNumString(a);
    SendString("\r\n");
```

```
}
```

对发送端号码解析，判断发送号码是否为授权用户，有效则执行短信内容（如对授权用户发送监测信息等），无效则删除短信。

```
char ReadTel( )   //读取授权用户号码，允许授权 5 个
{
char i;
if(byte_read(TEL1_POS) ==0x55)   //授权第 1 个号码
    {
        for(i =1;i <13;i ++ )
        tel1[i -1] =byte_read(TEL1_POS +i);
    }
    else
    {
        tel1[0] =0;
    }
    if(byte_read(TEL2_POS) ==0x55) //授权第 2 个号码
    {
        for(i =1;i <13;i ++ )
        tel2[i -1] =byte_read(TEL2_POS +i);
    }
    else
    {
        tel2[0] =0;
    }
    …
return 0;
}
```

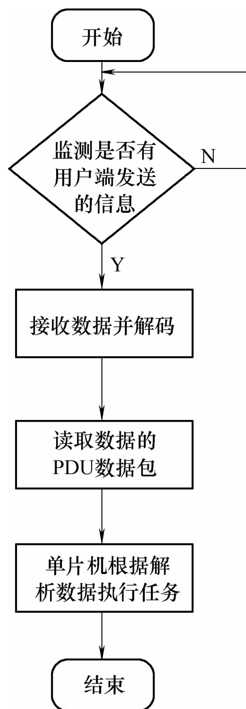


图 19-18　短信接收程序流程图

如果在设定的时间内用户没有反馈信息，则停止等待回到主程序中。若收到用户反馈信息，则通过“AT + CMGR”命令，读取短信息的 PDU 数据包，依据解析的代码执行相应的 AT 指令。

## 19.5　系统测试

### 19.5.1　系统测试步骤

用手机对采集感知端远程监测，测试步骤如下：

（1）授权号码测试　为保证系统网络安全运行，先授权合法用户。输入 cmd138XXXX8084，短信回复 Tel1 is OK！说明用户 1 授权成功，如继续输入“cmd + 用户手机号码”，短信回复 Tel2 is OK!，为了保障用户可以准确收到感知的果苗生长信息，系统最

多可设置 5 个授权用户。

（2）采集空气和土壤温湿度信息测试　发送短信 temp 到采集感知端，短信返回所有感知的果苗参数信息。Air T：空气温度，Air RH：空气湿度，Soil T：土壤温度，Soil RH：土壤湿度。

（3）报警测试　任一项果苗生长监测参数不在门限值范围内，除了现场端报警之外，系统同时将警情信号发送到用户端，返回短信为 ERROR！The value of XX is too high/low. XX 为 air temperature：空气温度；XX 为 air moisture：空气湿度；XX 为 soil temperature：土壤温度；XX 为 soil moisture：土壤湿度。

（4）号码删除测试　若用户想更换新的授权号码，发送短信 del 到采集感知端，则返回短信为 All tel removed！此时可以重新授权用户。

### 19.5.2　测试结果分析

选取一个采集感知端上电运行，用户依据 19.5.1 节的 4 个测试步骤顺序操作，手机端返回信息如图 19-19 所示，通过对 15904091469 用户进行授权，发送 temp 指令，将采集感知端监测信息以短信方式发送到用户端，显示当前空气温度 16°，空气湿度 30% RH，土壤温度 14°，土壤湿度 60% RH，接收该短信时间大约 10s 左右。为了便于分析，对采集感知端土壤传感器人工加湿使其湿度超过设定的 82% RH，现场电路报警，报警时间大约 25s，同时警情短信 ERROR！The value of soil moisture is too high 发送到用户手机端，告知用户该项监测数据超标，测试接收该警情短信时间大约 5s 左右，若重新授权用户则输入 del 命令即可。采集感知端运行系统如图 19-20 所示，整个系统运行稳定、工作良好，监测数据准确。




图 19-19　用户终端指令操作界面




图 19-20　采集感知端调试界面

## 19.6　结论

目前，国内果园生产管理主要靠人工完成，果园环境参数、果苗生长状况都是靠人为经验获取。这种依靠经验的生产模式已经远远不能跟上果园生产自动化进程，工作效率低下、监测误差比较大。给出的设计系统表明果农在任意时间、任意地点通过手机能够对果苗的生

长参数远程监测，若监测数据不在门限值范围内会短信告知果农，确保果农第一时间对果苗及时处理，从而促进果苗生长提高果实品质。系统也可以应用到农村蔬菜种植、花卉园艺等各种类型的温室大棚中，用于环境参数的监测，也可接入自动控制设备，将环境参数监测与自动控制联合使用，形成一个完整的智能化控制系统。

## 附件：果园现场数据采集端电路原理图

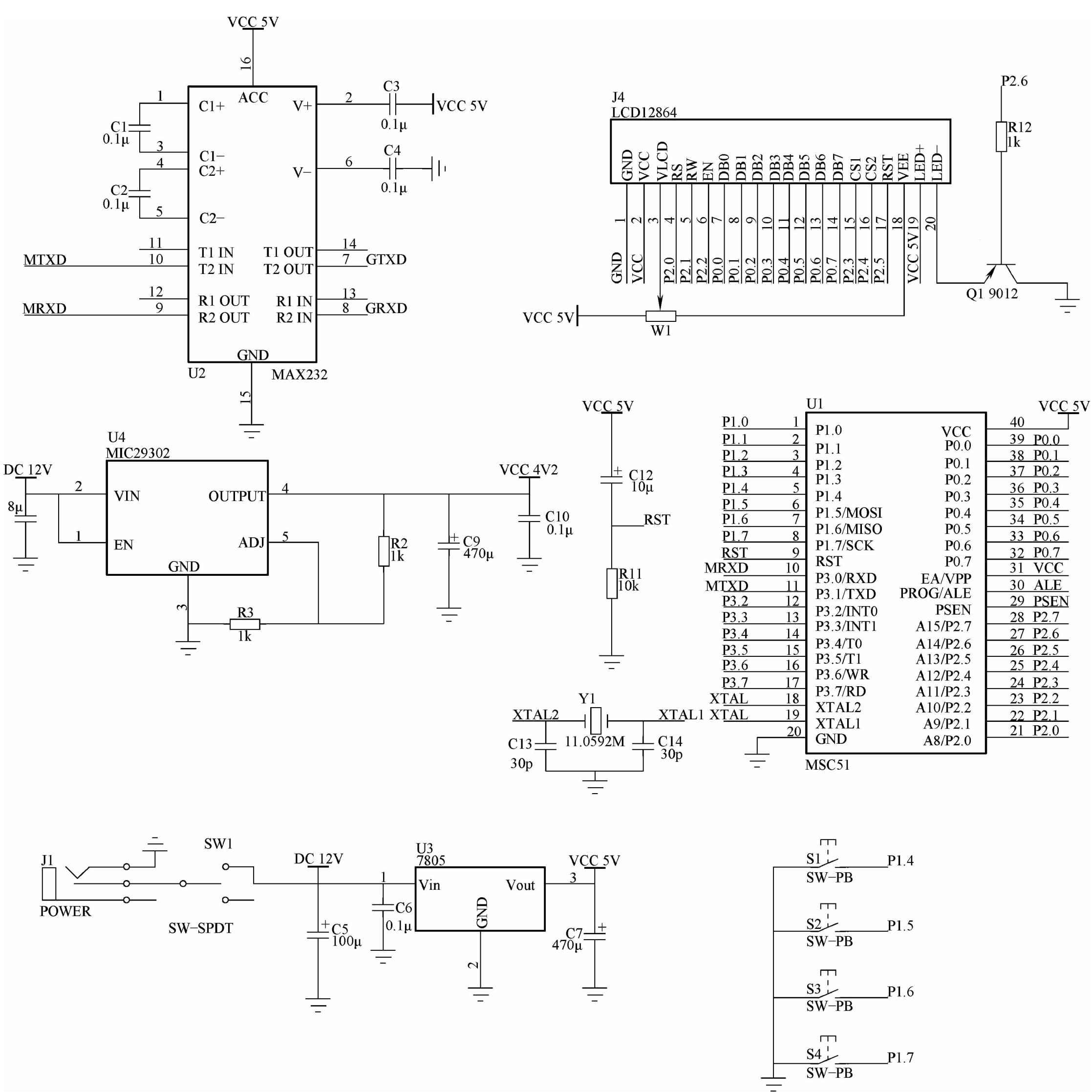



图 19-21 果园环境监测设计的电路原理图

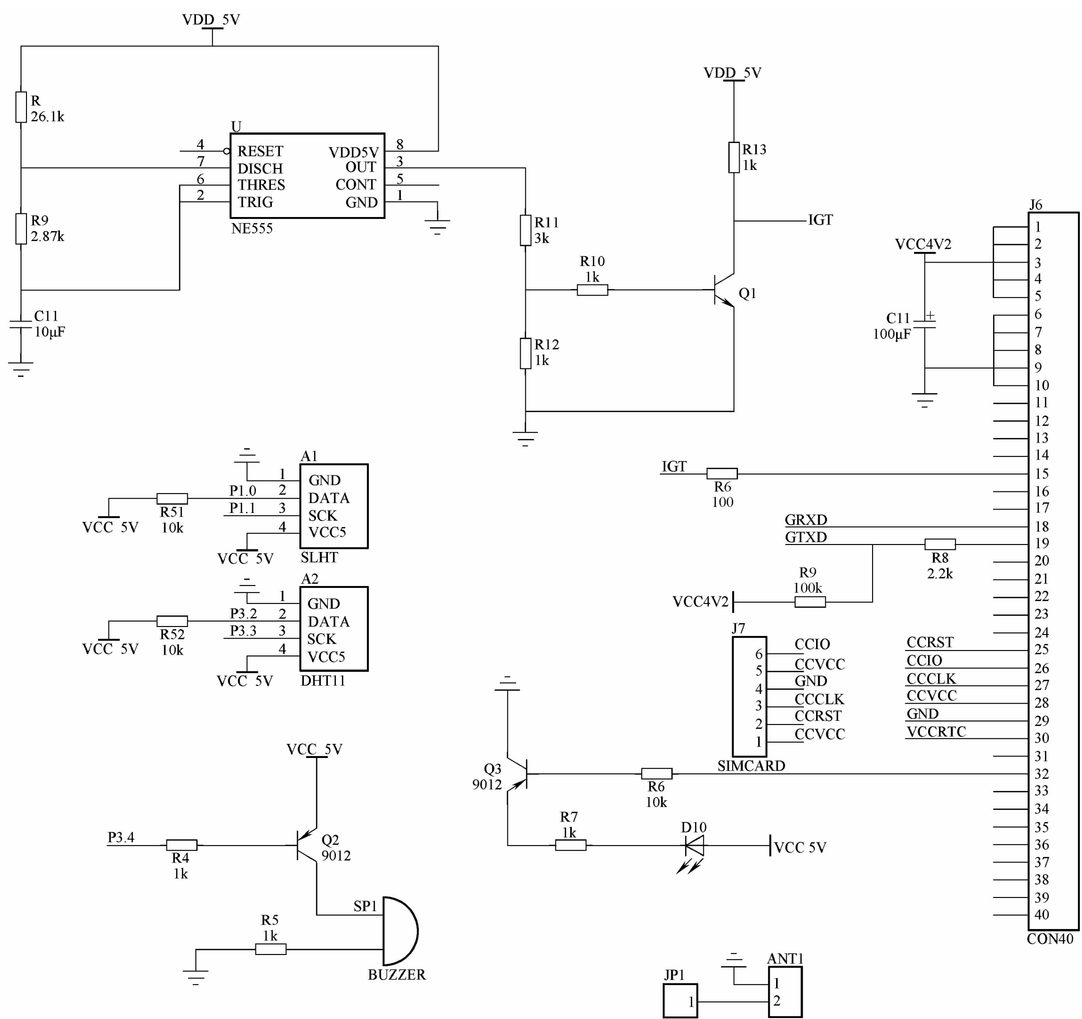



图 19-21　果园环境监测设计的电路原理图（续）

# 第 20 章　单片机实现电子密码锁设计

## 20.1　项目说明

### 20.1.1　项目背景

随着人们生活水平的提高，人们对日常生活中安全保险器件的要求越来越高，为满足人们对锁的使用要求，增加其安全性，用电子密码代替钥匙的密码锁应运而生。目前应用比较广泛的电子密码锁有遥控式电子密码锁、卡式电子密码锁、键盘式电子密码锁等，现对这几种常用形式的电子密码锁优缺点分析比较如下：

①遥控式电子密码锁：有红外遥控与可见光遥控，这两类密码锁的特点是保密性比较高、传输信息量大、速度快。

②卡式电子密码锁：有接触式与非接触式两种，它的优点是存储信息量大，可以一卡多用。对于①②上述两类电子密码锁，其缺点是都必须保存好遥控器或卡（钥匙）。

③键盘式电子密码锁：具有操作快、修改密码比较简单随意等优点，但密码不能过于简单或过于复杂。下面介绍当前几种形式的电子密码锁。

（1）遥控式电子防盗锁　目前常见的遥控式电子防盗锁主要有光遥控和无线电遥控两类。光遥控又分为红外线遥控和可见光遥控，光遥控利用窄角度的光传输密码，优点是传输信息量大、速度快、人眼识别不出来、无法在光路径上（操作者与电子防盗锁主体之间）以仪器捕获信号试图复制，因此保密性极高。无线电遥控的优点是传输信息量大、速度快、人眼识别不出来，但是发射的信号弥散空间，容易被仪器捕获，因此适合采用“变化的密码”，如所谓的“跳码”、“滚码”（均是随机变化而无明显规律），这样即使捕获了当时的信号也无利用、复制的价值。使用遥控式电子防盗锁，需要仔细保管遥控器（即钥匙），而且对某些应用而言，这种钥匙大了一点，可能还要使用特定的电池。

（2）卡式电子防盗锁　使用各种“卡”作为钥匙的电子防盗锁是当前最为活跃的产品，无论卡的种类如何多种多样，按照输入卡的操作方式，都可分为接触式卡和非接触式卡两大类。目前接触式卡的技术成熟、价格较低，应用也较为广泛；非接触式卡使用隐蔽、方便，大有后来居上之势。储存信息量大是卡的优势，它不仅作为钥匙，还可载入多项个人信息，特别适合金融业注重“验明正身”的行业特点，而且一卡多用（如入门、开锁、存储、付费等）带来持卡人的便利。使用这类电子防盗锁，需要仔细保管卡（即钥匙），尤其丢失了必须尽快取消该卡的授权。

（3）键盘式电子密码锁　从目前的技术水平和市场认可程度看，使用最为广泛的是键盘式电子密码锁，该产品主要应用于保险箱、保险柜和金库，还有一部分应用于保管箱和运钞车。键盘式电子密码在键盘上输入，其突出优点是“密码”是记在被授权人脑子里的数字和字符，既准确又可靠，不会丢失（除了忘记），难以被窃（除非自己泄露）。但是密码

不能太简单，太简单了就容易被他人在键盘上试探出来，或者可能被旁观者窥测出来，造成保密性不足。当然，密码又不能太复杂，太复杂了可能自己都糊涂了，或者输入密码操作成功率低，造成使用不便。因此，为了发扬优点、克服弱点，键盘式电子密码也在不断发展中，如“任意设定密码”技术使得被授权人可以根据自己的需要或喜好设定密码；而“自动更改密码”技术使得本次输入的密码将自动更改成下次应输入的密码，更改的规律不为他人所知，因而不怕旁观者窥测；独出心裁的“键盘乱序显示”技术使得键盘上的固定键位每次显示出的字符不固定，并且显示的窄小角度只能由操作者正面看得到，因而即使旁观者看见操作动作也难以窥测出密码；“多重密码设定”技术使得单组密码不一定有效，适合多人分权使用，需要输入两组以上的密码才被认可，大大提高了保密性，如果限定输入这些密码的先后顺序或时间区段，则保密性还可提高。在输入密码的过程中，为了限制试探密码的企图，通常输入错误码若干次或若干时间内输入不正确，即“封锁”键盘，不再接受输入操作。总之，尽管新式电子防盗锁层出不穷，但键盘式电子密码防盗锁仍然“老树发新芽”，不仅在市场上居于主流地位，而且，还经常作为其他类型电子防盗锁的辅助输入手段。

由于数字、字符、图形图像、人体生物特征和时间等要素均可成为钥匙的电子信息，所以电子密码锁还是目前的主流产品。

### 20.1.2　电子密码锁优点

项目的主要任务是设计一款基于单片机技术的电子密码锁，该电子密码锁具有以下优点：

①采用键盘式电子密码锁技术，具有灵活修改密码、设置密码、成本低、功耗低等优点。

②键盘式电子密码锁技术具有安全性高，可方便地应用在各种安全性要求高的场合。

③密码存储采用 $E^2$PROM 芯片 AT24C02，具有密码信息不丢失、接线简单、成本低廉等优点。

设计的电子密码锁，具有一定的工程意义和市场应用价值，尤其应用在一些对安全性要求比较高的系统或机构中。

### 20.1.3　研究内容

电子密码琐需要实现功能如下：

①密码输入：每按下一个数字键，就输入一个数值，并在显示器上的最右方显示出该数值，同时将先前输入的数据依序左移一个数字位置。

②密码清除：按下此键可清除键入的值。

③密码修改：如果密码验证正确，可进行密码修改，将拨动开关拨至左边（初始化为右边），按下此键，可进入密码修改状态，输入 6 位密码，按下确认键，LED 亮，说明密码修改成功，且在数码管上显示 6 个“8”。

④密码确定：按下此键会检查输入的密码是否正确，如果正确，数码管显示“HELLO”蜂鸣器响一下，LED 亮一下，如果错误，则显示“ERROR”，蜂鸣器响四下，LED 亮四下。

⑤密码返回：按下此键可重新输入新密码。

# 20.2 系统总体设计

## 20.2.1 系统工作原理

系统设计用6位数码管组成显示电路提示信息，接着输入6位数的密码，在数码管上显示6个“8”，当密码位数输入完毕按下确认键后，输入的密码与设定的密码进行比较，若密码正确，则数码管显示“HELLO”，蜂鸣器响一下，LED亮一下；如果错误，则显示“ERROR”，蜂鸣器响四下，LED亮四下；如果密码正确，也可进入密码修改状态，先将拨动开关拨到左边，然后按下修改键，就可以进行密码修改，输入6位密码，按下确认键，如果指示灯亮，则表明密码修改成功，在数码管上显示6个“8”，且将新的密码存入串行EEPROM中；在密码输入过程中，若输入错误，可以利用“DEL”键删除刚才输入的错误数字。如果按“ESC”键则全部删除。

## 20.2.2 系统结构

本系统利用单片机Atmel公司的AT89S52作为核心控制元件，其外围电路整体规划上主要分为四个模块：密码存储模块、数码管和LED显示模块、键盘输入模块、蜂鸣器报警模块，系统结构框图如图20-1所示。

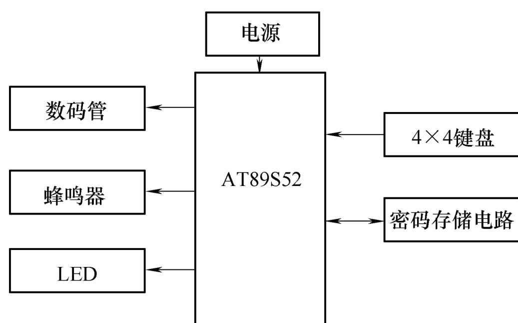


图20-1 系统结构框图

密码存储模块是本系统的关键模块，主要功能是实现密码的保存，本系统中密码存储采用带$I^2C$总线的串行$E^2PROM$芯片AT24C02。该芯片具有如下优点：

①简化了硬件电路，在这种总线中只需要两根线，即串行数据线和串行时钟线。

②总线接口协议中有冲突监测和仲裁机制，以防止通信中的数据丢失或发生错误。

③这种串行的两线双向总线在标准模式下的速率为100kbit/s，在快速模式下的速率为400kbit/s，在高速模式下为3.4Mbit/s。

④器件中有滤波抗扰措施来保证数据的完整性。

数码管显示模块主要是用于显示密码代替符“8”，输入几个密码就显示几个“8”，最多只能输入6位数的密码值；当密码输入正确的前提下，再按确认键的时候，数码管上会显示“HELLO”，如果错误的话，则显示“ERROR”。

键盘模块是本系统中的主要输入模块，主要功能有输入密码、确认密码、密码清除、密码删除和密码修改等功能，这些按键功能的增加有效地完善了键盘的功能，使键盘能够更有效地工作，更能保证键盘的保密性和可靠性。

蜂鸣器报警模块主要用于在密码输入错误的前提下，再按下确认键的时候，蜂鸣器会响四声。

# 20.3　系统硬件电路设计

该电路选用 AT89S52 单片机作为 MCU，选用 11.0592MHz 晶振以及两个 30pF 电容作为时钟电路，给系统工作提供基本的时钟基准信号，选用手动加上电复位作为系统的复位电路，使得电路在发生异常时可随时复位回到初始状态。选用 $I^2C$ 总线技术的串行 $E^2PROM$ 芯片 AT24C02 作为密码存储芯片；选用 4×4 矩阵键盘作为密码输入的非编码式键盘；选用 74LS07 作为驱动器和 6 个八段共阴数码管作为显示模块，作为电路的显示模块；选用蜂鸣器与一个晶体管构成蜂鸣器报警模块；选用 7805 变压器作为系统电源供电模块。

## 20.3.1　AT89S52 单片机最小系统设计

AT89S52 是一种低功耗、高性能 CMOS8 位微控制器，具有 8KB 在系统可编程 Flash 存储器。使用 Atmel 公司高密度非易失性存储器技术制造，与工业 80C51 产品指令和引脚完全兼容。片上 Flash 允许程序存储器在线系统编程，亦适于常规编程器。在单芯片上，拥有灵巧的 8 位 CPU 和系统可编程 Flash，使得 AT89S52 为众多嵌入式控制应用系统提供高灵活、有效的解决方案。

AT89S52 单片机具有如下功能：

①与 MCS-51 单片机产品兼容。

②8KB 在线系统可编程 Flash 存储器。

③1000 次擦写周期。

④全静态操作：0～33Hz。

⑤三级加密程序存储器。

⑥32 个可编程 I/O 口线。

⑦三个 16 位定时/计数器。

⑧八个中断源。

⑨全双工 UART 串行通道。

⑩低功耗空闲和掉电模式。

⑪掉电后中断可唤醒。

⑫看门狗定时器。

⑬双数据指针。

⑭掉电标识符。

单片最小系统有晶振电路和复位电路构成，复位电路采用手动复位高电平有效。其最小系统如图 20-2 所示。

本系统设计中单片机的资源分配具体如下：采用 P0 口输出控制数码管的段码信号；P2 口输出控制数码管的位码选通信号；P3 口上半口输出作为键盘的行控制线，下半口输入作为键盘的列状态线；P1.2，P1.3 作为连接存储器的 SDA，SCL 两根信号线，控制存储器的读写操作；P1.0，P1.1 作为输出口控制 LED 的亮灭；P1.5 作为输出口控制密码修改时的一个操作；P1.7 作为输出口控制蜂鸣器的发声。

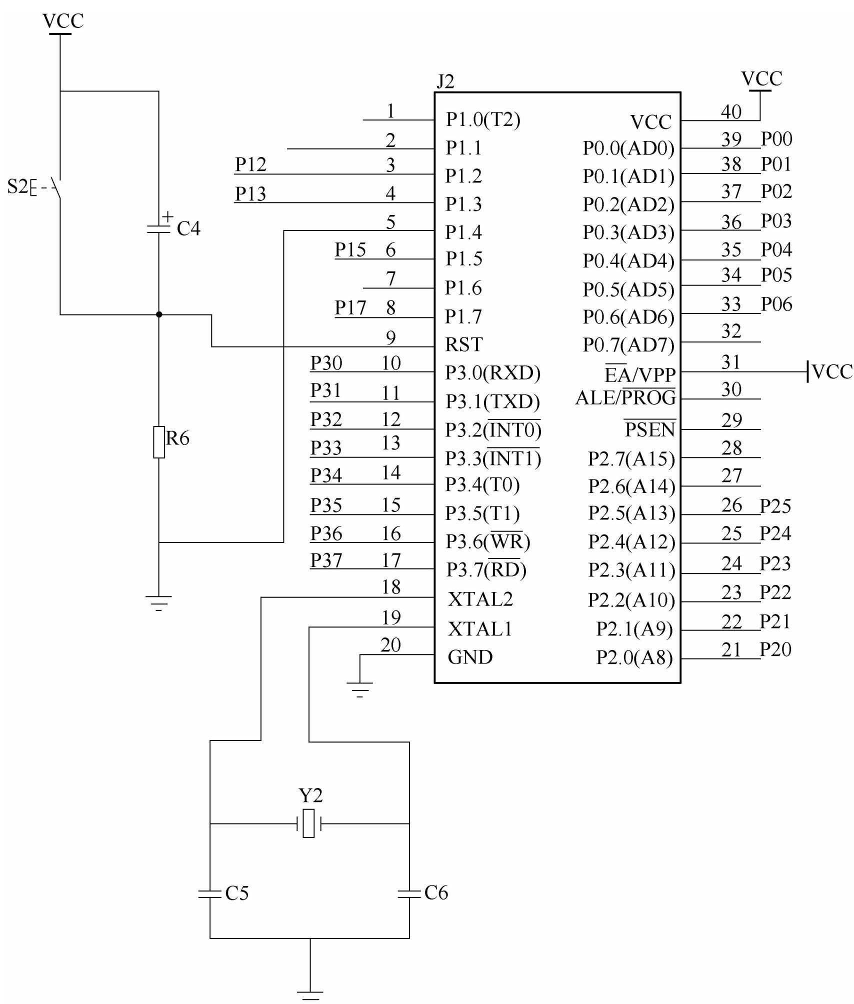



图 20-2　单片机最小系统电路

## 20. 3. 2　密码存储电路设计

AT24C02 是美国 Atmel 公司的低功耗 CMOS 型 $E^2$PROM，内含 256 × 8 位存储空间，具有工作电压宽（2. 5 ~5. 5V）、擦写次数多（大于 10000 次）、写入速度快（小于 10ms）、抗干扰能力强、数据不易丢失、体积小等特点。而且它是采用了 $I^2C$ 总线式进行数据读写的串行器件，占用很少的资源和 I/O 线，并且支持在线编程，进行数据实时的存取十分方便。

AT24C02 的引脚图如图 20-3 所示，该芯片的引脚功能具体如下：它的 1、2、3 脚是 3 根地址线，用于确定芯片的硬件地址；第 8 脚和第 4 脚分别为正、负电源；第 5 脚 SDA 为串行数据输入/输出，数据通过这根双向 $I^2C$ 总线串行传送；第 6 脚 SCL 为串行时钟，SDA 和 SCL 为漏极开路端，在实际的应用当中都需要和 VCC 间各接一个 5. 1kΩ 的电阻上拉；第 7 脚为 WP 写保护端，接地时允许芯片执行一般的读写操作，接电源时只允许对器件进行读操作。

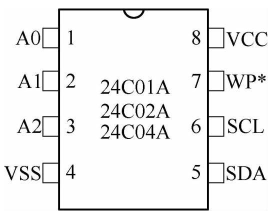



图 20-3　AT24C02 芯片引脚图

引脚说明：

SCL：串行时钟。在该引脚的上升沿时，系统将数据输入到每个 EEPROM 器件，在下降沿时输出。

SDA：串行数据。双向串行数据/地址引脚用于器件所有数据的发送或者接收。SDA 是一个开漏输出引脚，可以与其他开漏输出或者集电极开路输出进行线或。

A0、A1、A2：器件地址输入端。用于多个器件级联时设置器件地址，引脚悬空时默认值为 0，如果只有一个 2402 被总线寻址，三个地址输入引脚接到 VSS 地端。

WP：硬件写保护。当该引脚为高电平时禁止写入，所有内容都被写保护。当为低电平时可正常读写数据。

VCC：电源。一般输入 5V 电压。

VSS：接地。

24C02 支持 $I^2C$ 总线传输协议，$I^2C$ 是一种双向、两线串行通信接口，分别是串行数据线 SDA 和串行时钟线 SCL。总线上发送数据的器件为发送器，接收数据的器件为接收器，控制信息交换的器件为主器件，受主器件控制的器件则为从器件，主器件产生串行时钟 SCL，控制总线的访问状态，产生 START 和 STOP 条件，24C02 在 $I^2C$ 总线中作为从器件工作。只有当总线处于空闲状态时才可以启动数据传输，每次数据传输均开始于 START 条件，结束于 STOP 条件，两者之间的数据字节数是没有限制的，由总线上的主器件决定，信息以字节（8 位）为单位传输，第 9 位时由接收器产生应答信号，24C02 存储电路如图 20-4 所示。

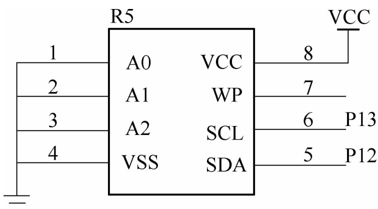


图 20-4　24C02 存储电路

### 20.3.3　4×4 矩阵键盘模块设计

本系统设计将键盘分为两大模块，分别是数字键模块和功能键模块。数字键模块是包括了从 0~9 的 10 个数字；功能键模块包括了修改、返回、删除、确认这些功能按键，每个按键分别完成相应的功能。键盘结构如图 20-5 所示。

在矩阵式键盘中，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接。这样，一个 P3 口就可以构成 4×4 = 16 个按键，比直接将端口线连接键盘多出了一倍，而且线数越多，区别越明显，如再多加一条线就可以构成 20 键的键盘，矩阵键盘电路原理图如图 20-6 所示。

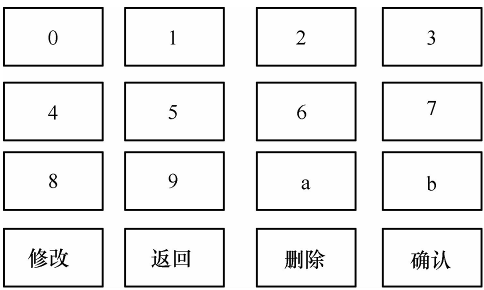


图 20-5　键盘结构图

矩阵式结构的键盘显然比独立按键要复杂，识别也要复杂一些，图 20-6 中，行线接单片机的 I/O 口作为输出端，而列线所接的 I/O 口则作为输入端。这样，当按键没有按下时，所有的输出端都是高电平，代表无键按下。行线输出是低电平，一旦有键按下，则输入线就会被拉低，这样，通过读入输入线的状态就可得知是否有键按下了。

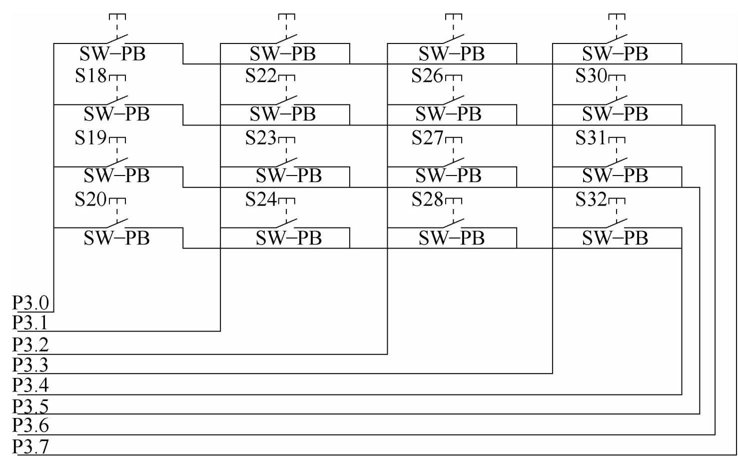



图 20-6　矩阵键盘电路原理图

## 20. 3. 4　数码管显示电路设计

数码管由 8 段发光二极管组成，成一个日字形，它们可以是共阴极的，也可以是共阳极的。每个数码管都有 a、b、c、d、e、f、g 7 个笔画和 1 个小数点 dp，这 8 个对应二极管阳极都连在一起（称共阳极），阴极都连在一起（称共阴极）。数码管可以通过硬件解码电路得到相应的段码，也可以通过软件查询段码表得到相应数字的段码。本系统设计时采用 P0 控制数码管的段码，而 P2 口控制数码管的位码。P2 口与数码管之间连接是通过数码管驱动芯片 74LS07 实现，6 个数码管显示电路原理图如图 20-7 所示。

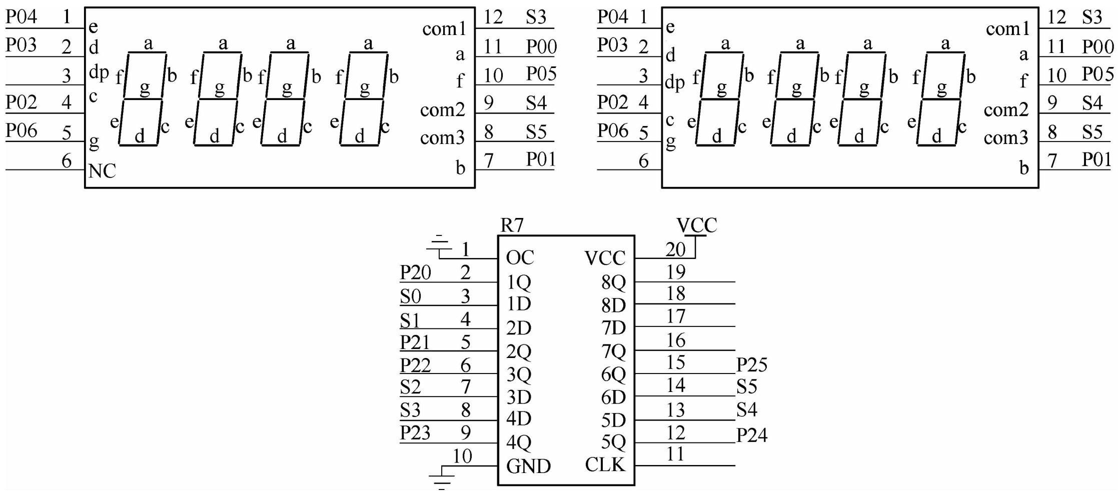



图 20-7　数码管显示电路

## 20. 3. 5　报警指示模块设计

用单片机的 P1. 7 口作为输出口控制蜂鸣器发声。蜂鸣器的最重要的一个特点是只要按照极性要求加上合适的直流电压，就可以发出固有频率的声音。具体的电路图如图 20-8 所示：

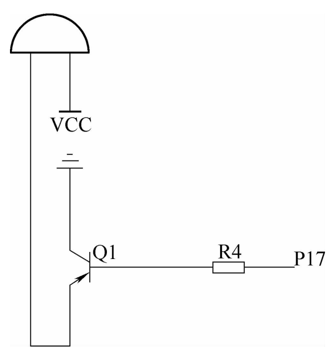



图 20-8　蜂鸣器报警电路

### 20.3.6　电源模块电路设计

直流电源采用整流、滤波、稳压，具体电路如图 20-9 所示。普通电源插座的后级采用 4 个二极管组成全桥整流电路，外接输入电压范围是交流 8～12V；整流后采用了 470μF 大容量电解电容进行滤波，同时增加了高频滤波去耦电容；稳压电路采用 7805 三端稳压集成电路，同时配备了专业的铝散热片，确保三端稳压集成电路 7805 的良好散热效果，输出的电源电压更稳定，纹波更低。

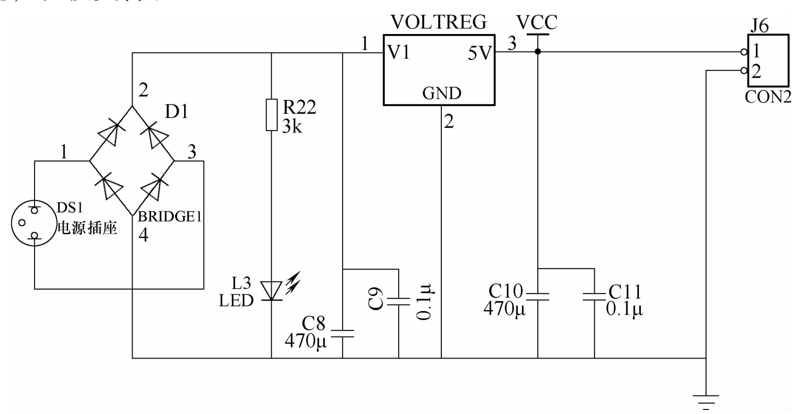



图 20-9　电源供电电路

7805 为 3 端正稳压电路，能够提供多种固定输出电压，应用范围广。内部包含过电流、过热和过载保护电路。带散热片时，输出电流可达 1A，虽然是固定稳压电路，但使用外接元件，可获得不同的电压和电流。7805 实物如图 20-10 所示。其芯片的引脚说明如下：

1 脚：输入端；2 脚：接地端；3 脚：输出端。

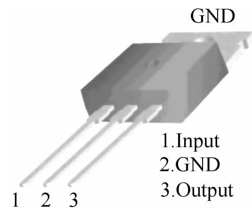



图 20-10　7805 实物图

## 20.4 系统软件程序设计

软件设计主要采用 Keil 仿真编译环境，用 C 语言编写来完成矩阵键盘电子密码锁的程序设计，其主要完成的任务有修改密码、保存密码、识别密码、出错报警以及基于 $I^2C$ 总线技术的密码存储设计。

本系统软件设计分为以下几个模块：主程序、串行 EEPROM 读写程序、数码管动态显示、4×4 矩阵键盘处理以及蜂鸣器报警程序，下面对各模块工作软件流程设计介绍如下。

### 20.4.1 主程序设计

系统上电，主程序开始扫描键盘，此时，数码管上不显示任何数据。若有键按下，则判断该键值，若为数字键，则认为是密码输入，数码管显示“8”；若为确认键，则程序开始核对输入的数据是否和设定的密码数据相同，相同则 D1 指示灯闪一下，蜂鸣器响一声，不同则 D1 指示灯闪四下，蜂鸣器响四声；若为返回键，则清除所输入的密码，此时数码管全灭；若为清除键，则清除最后输入的一位密码，数码管显示数据中输入的最后那一位将被清除；若为修改键，则再输入正确后，按此键则可输入新密码。其程序流程图如图 20-11 所示。

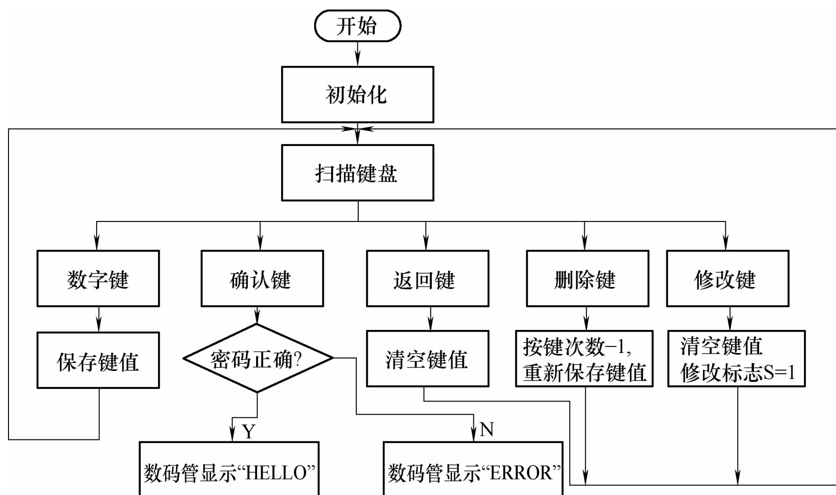



图 20-11 主程序流程图

主函数初始化，如果 P1.4 口接低电平，将读取 24c02 EEPROM 中的密码值放入 ps2［6］数组中代码如下：

```
if(k1 ==0)
{   for(i =0;i <6;i ++)
      {
       x24c02_init();
      inips =x24c02_read(i);
```

```
        ps2[i] = inips;        }
    }
```

密码正确且 P1.5 =0 且 key ==12，进入密码修改状态，并显示数据。

```
    if(k2 ==0&&tt ==1&&s ==1)
      for(b =0;b <6;b ++)
        {   P0 = dispcode[data3[b]];
          P2 = tem;
          tem =0x01;
          tem = tem <<(b +1);
          delay(300);
        }
        …
```

密码正确且 key =12，进入密码修改状态，新密码存入 data3[6]数组中。

```
    if(k2 ==0&&tt ==1&&s ==1)
        { data2[cnt] = cunchu(key,cnt);
          data3[cnt] =16;        }
            else                          //输入初始密码,存入 data1[6]数组中
        {   data1[cnt] = key;
            data4[cnt] =16;   }
              cnt ++;
        if(cnt >6)
          {   cnt =6;
            overflag =1;
          }
```

## 20.4.2　串行 EEPROM 读写程序设计

由于 AT89S52 单片机内部没有集成 $I^2C$ 总线模块，采用 C 语言编写 $I^2C$ 总线技术的通信协议，实现单片机与 AT24C02 存储器的密码读出与写入功能。

**1. $I^2C$ 总线的传输协议和数据传输**

（1）起始和停止条件

在数据传送过程中，必须确认数据传送的开始和结束。在 $I^2C$ 总线技术规范中，开始和结束信号（也称启动和停止信号）的定义如图 20-12 所示。

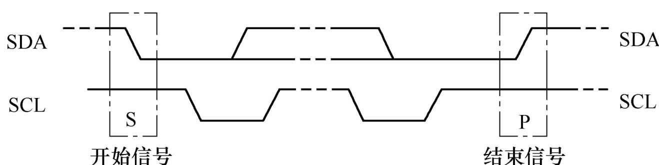



图 20-12　超始和停止信号图

开始信号：当时钟总线 SCL 为高电平时，数据线 SDA 由高电平向低电平跳变，开始传送数据。

结束信号：当 SCL 线为高电平时，SDA 线从低电平向高电平跳变，结束传送数据。

开始和结束信号都是由主器件产生。在开始信号以后，总线即被认为处于忙状态，其他器件不能再产生开始信号。主器件在结束信号以后退出主器件角色，经过一段时间后，总线被认为是空闲的。

（2）数据格式

$I^2C$ 总线数据传送采用时钟脉冲逐位串行传送方式，在 SCL 的低电平期间，SDA 线上高、低电平能变化，在高电平期间，SDA 上数据必须保护稳定，以便接收器采样接收，数据传送时序如图 20-13 所示。

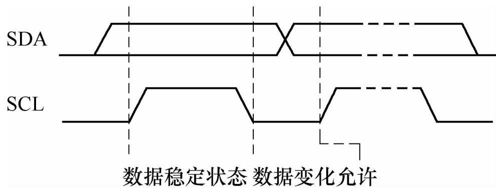



图 20-13　数据传送时序图

$I^2C$ 总线发送器送到 SDA 线上的每个字节必须为 8 位长，传送时高位在前，低位在后。与之对应，主器件在 SCL 线上产生 8 个脉冲；第 9 个脉冲低电平期间，发送器释放 SDA 线，接收器把 SDA 线拉低，以给出一个接收确认位；第 9 个脉冲高电平期间，发送器收到这个确认位然后开始下一字节的传送，下一个字节的第一个脉冲低电平期间接收器释放 SDA。每个字节需要 9 个脉冲，每次传送的字节数是不受限制的。

$I^2C$ 总线的数据传送格式是在 $I^2C$ 总线开始信号后，送出的第一字节数据是用来选择从器件地址的，其中前 7 位为地址码，第 8 位为方向位（R/W）。方向位为“0”表示发送，即主器件把信息写到所选择的从器件中；方向位为“1”表示主器件将从从器件读信息。格式如下：

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | R/W |

注：前四位固定为 1010。

开始信号后，系统中的各个器件将自己的地址和主器件送到总线上的地址进行比较，如果与主器件发送到总线上的地址一致，则该器件即被主器件寻址的器件，其接收信息还是发送信息则由第 8 位（R/W）决定。发送完第一个字节后再开始发数据信号。

（3）响应

数据传输必须带响应。相关的响应时钟脉冲由主机产生，当主器件发送完一字节的数据后，接着发出对应于 SCL 线上的一个时钟（ACK）认可位，此时钟内主器件释放 SDA 线，一字节传送结束，而从器件的响应信号将 SDA 线拉成低电平，使 SDA 在该时钟的高电平期间为稳定的低电平。从器件的响应信号结束后，SDA 线返回高电平，进入下一个传送周期。通常被寻址的接收器在接收到的每个字节后必须产生一个响应。当从机不能响应从机地址时，从机必须使数据线保持高电平，然后主机产生一个停止条件终止传输或者产生重复起始条件开始新的传输。如果从机接收器响应了从机地址但是在传输了一段时间后不能接收更多数据字节，主机必须再一次终止传输。这个情况用从机在第一个字节后没有产生响应来表示。从机使数据线保持高电平主机产生一个停止或重复起始条件。完整的数据传送过程如图 20-14 所示。

$I^2C$ 总线还具有广播呼叫地址，用于寻址总线上所有器件的功能。若一个器件不需要广

播呼叫寻址中所提供的任何数据，则可以忽略该地址不作响应。如果该器件需要广播呼叫寻址中按需提供的数据，则应对地址作出响应，其表现为一个接收器。

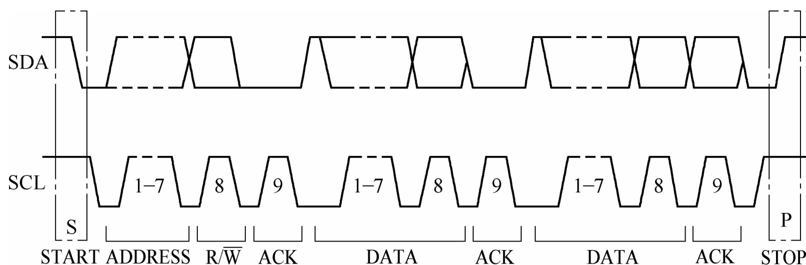



图 20-14　完整的数据传送过程

**2. AT24C02 读写操作**

本系统设计中利用 P1.2，P1.3 作为 SDA、SCL 串行通信线，实现对 AT24C02 芯片内某一地址数据的读写操作。2402 读写关键代码如下：

从 24c02 的地址 address 中读取一个字节数据

```
uchar x24c02_read(uchar address)
{   uchar i;
    start(); writex(0xa0);
    clock(); writex(address);
    clock(); start();
    writex(0xa1); clock();
    i = readx(); stop();
    delay1(10);
    return(i);
}
```

向 24c02 的 address 地址中写入一字节数据 info

```
void x24c02_write(uchar address,uchar info)
{   start(); writex(0xa0);
    clock(); writex(address);
    clock(); writex(info);
    clock(); stop();
    delay1(50);
}
```

对于存储器存储密码，为了方便调试，如果忘记存储器里的密码，此时 P1.4 引脚断开，连接 P1.5 引脚使其设置成高电平，软件程序中设置初始密码 123456，按下确认键，数码管显示“HELLO”；如果修改密码，可以把 P1.4 引脚接地，P1.5 引脚设置成低电平，输入新的密码存入存储器中，如果存储成功，L4 灯亮；下次只要把 P1.5 引脚设置高电平，P1.4

引脚接地，将输入的键值与存储器里的密码进行对比分析。

### 20.4.3　4×4 矩阵键盘处理程序设计

根据键盘的工作原理，本系统的键盘扫描方法采用行扫描方法。行扫描法又称为逐行（或列）扫描查询法，是一种最常用的按键识别方法，介绍过程如下：

（1）判断键盘中有无键按下　将全部行线置低电平，即 P3.0 ~ P3.3 输出全“0”，然后检测列线的状态，即读取 P3.4 ~ P3.7 的状态。只要有一列的电平为低，即 P3.4 ~ P3.7 中有一个“1”，则表示键盘中有键被按下，而且闭合的键位于低电平线与 4 根行线相交叉的 4 个按键之中。若所有列线均为高电平，即 P3.4 ~ P3.7 为全“1”，则键盘中无键按下。

（2）去除键抖动　为了保证键每闭合一次 CPU 仅作一次处理，必须去除键释放时的抖动。本文采用软件延时法，即当检测到有键按下后，延时一段时间，重新扫描键盘有无键按下，确认有键按下后再做下一步的检测判断。

（3）判断闭合键所在的位置　在确认有键按下后，即可进入确定具体闭合键的过程。其方法是依次将行线置为低电平，即在置某根行线为低电平时，其他线为高电平。在确定某根行线位置为低电平后，再逐行检测各列线的电平状态。若某列为低，则该列线与置为低电平的行线交叉处的按键就是闭合的按键。由此得到闭合键的行值和列值，然后可采用计算法或查表法将闭合键的行值和列值转换成所定义的键值。

（4）等待按键释放　为了保证按键每闭合一次 CPU 仅作一次处理，必须等待按键释放后再作下一次按键的判断和处理。键盘处理子程序流程图如图 20-15 所示。

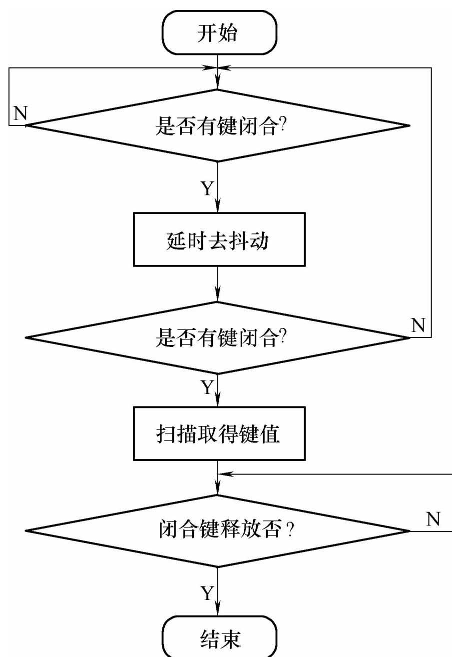



图 20-15　键盘处理程序流程图

键盘扫描关键代码如下：

```
uchar scankb(void)
{uchar sccode,recode;
P3 =0xf0;
if((P3&0xf0)! =0xf0)//有键按下
  {delay(1000);
  if((P3&0xf0)! =0xf0)//仍然有键按下
    {sccode =0xfe;
    while((sccode&0x10)! =0)//移位没完
      {P3 = sccode;//行扫描开始
      if((P3&0xf0)! =0xf0)//若在该行
        {recode =(P3&0xf0)|0x0f;//中间结果
        P3 =0xff;                    //关 P2
        return((~sccode)+(~recode));//返回 Keyword    }
      else sccode =(sccode <<1)|0x01;//不在该行则扫下一行   }
    }
  }
return(0);
}
```

## 20.5　系统调试总结

调试中主要的问题是怎么样的思路设计能让用户再忘记密码的时候可以开锁。其解决方法就是让与单片机的 P1.4 口相连的导线的接地与断开两种情况来控制，如果在密码验证正确的前提下，如果要修改密码的话，再加入一个参数，就是与 P1.5 相连的拨动开关的高低电平来控制密码的存储，如果忘记密码的话，可以使 P1.4 引脚线断开，也就是不与地相连，这个情况下，可以输入软件设定的初始密码 123456，即可开锁，如果要修改密码，即写新的密码值进存储器中，此时只要按下修改键，再把与 P1.5 相连的拨动开关拨至低电平，这个时候用户就可以输入新的密码存进存储器中，以后要开锁的话，就把 P1.4 接地，P1.5 的拨动开关拨至高电平，这样就可以读出存储器里的密码。

## 附件：系统设计电路原理图

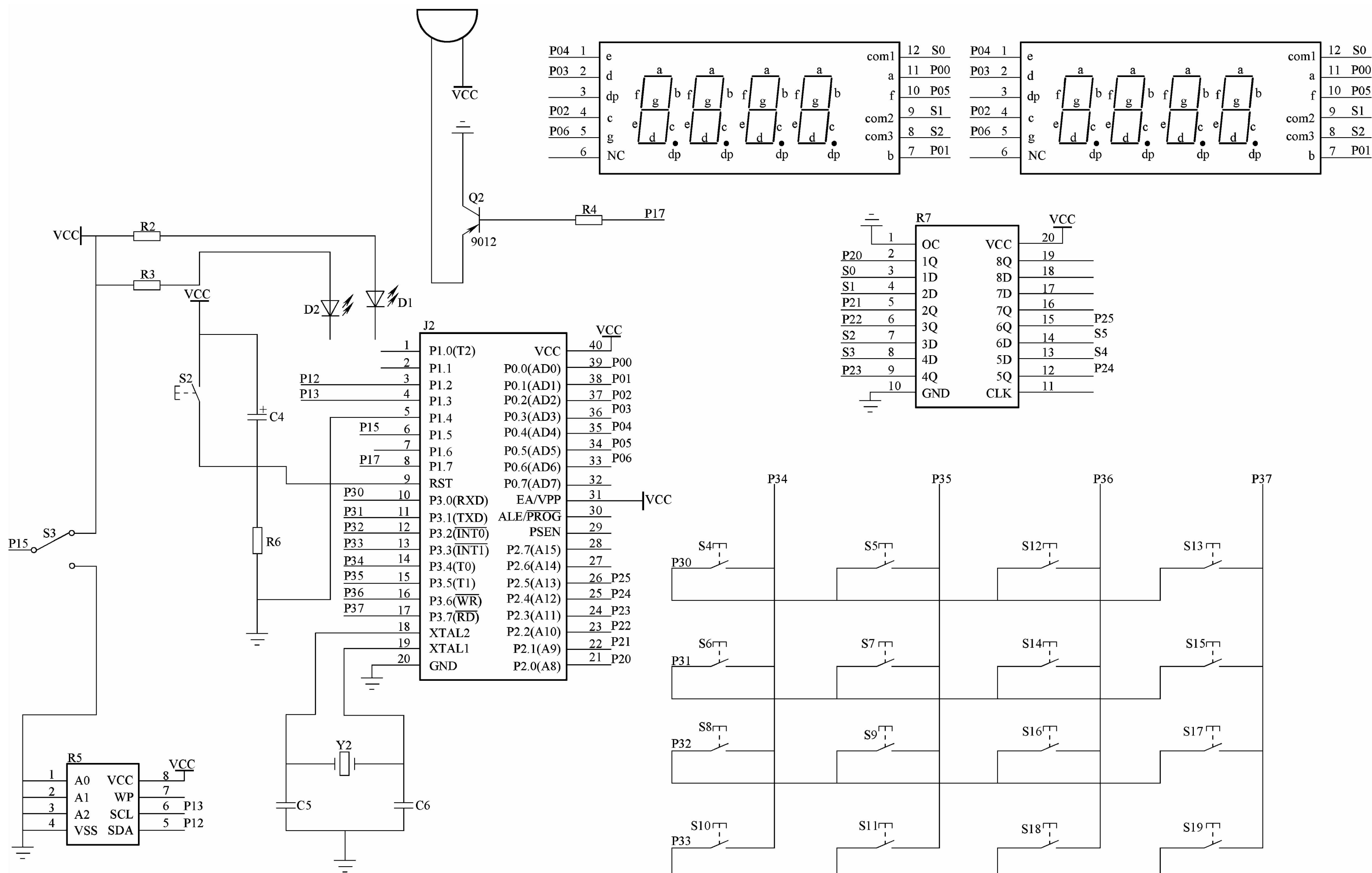



图20-16　系统设计电路原理图

# 第 21 章　红外遥控电动机转速系统设计

## 21.1　项目说明

### 21.1.1　研究背景

红外线遥控是目前使用最广泛的一种通信和遥控手段，红外遥控是一种无线、非接触控制方式实现对控制系统的操纵，不影响周边环境、不干扰其他电器设备，编解码容易，可进行多路遥控。在一些比较恶劣的环境中使用遥控器，可以保护操作者的安全。具有抗干扰能力强、信息传输可靠、功耗低、成本低、易实现等显著优点，所以红外遥控系统在工业自动化系统、高级家用电器、空调系统和其他方面已获得广泛应用。由于红外遥控应用范围广，根据不同的装置系统，红外遥控的功能不一样。按照遥控功能的复杂程度，分为在红外遥控发射器上有采用液晶（LCD）或者发光数码管（LED）显示器和不采用显示器两种机型。如彩色电视机的红外遥控发射器一般不采用显示器，对电源开关，选台，对比度，音量大小等各种功能均可实现红外遥控。录象机的红外遥控发射器都采用 LCD 显示器，空调系统采用 LCD 或 LED 显示器。

对于不同装置的红外遥控系统，其遥控的工作模式不一样，但红外遥控的工作原理大致相同。红外遥控系统一般采用不同的编码信号通过发射、接收放大和信号处理来实现不同的功能控制。不论是发射器或是主机板，均采用大规模集成电路 CPU 来实现各种遥控功能编码信号的发射和接收信号的解码处理，同时发射和接收部分一定要采用相对应的软件编程。红外遥控发射器一般采用 CPU 按照各种功能要求进行编码，这些不同的编码信号通过晶体管放大后驱动红外管，编码形式以红外信号发射，主机部分的接收红外编码信号一般采用 PIN 结构的红外光敏管，通过放大，检波后将编码信号送入主机板的 CPU 进行解码处理。并有相对应的软件编程和存储部分，解码处理后的信号通过接口部分分别控制实现各种不同的功能，来完成遥控系统的全过程。

目前，单片机的应用十分广泛，在工业控制领域、家电产品、智能化仪器仪表、计算机外部设备，特别是机电一体化产品中，都有重要的用途。其主要的用途可以分为以下方面。

1）显示：通过单片机控制发光二极管或是液晶，显示特定的图形和字符。

2）机电控制：用单片机控制机电产品做定时或定向的动作。

3）检测：通过单片机和传感器的联合使用，用来检测产品或者工况的意外发生。

4）通信：通过 RS-232 串行通信或者是 USB 通信，传输数据和信号。

本系统以单片机为处理核心，采用软件译码方法和抗干扰技术对遥控信号进行分析判断和解码，实现了对步进电机的控制。本系统红外线接收部分采用的是一体化接收头，将信号解调和放大全部集成在一起，提高了可靠性。这样，接收头送到单片机的就是编码的数据信号，而不是调制信号，数据的解码通过单片机完成。

### 21.1.2 研究内容

项目研究目标是采用单片机技术设计一个红外遥控的电机转速系统，并完成其硬件电路设计与软件程序的实现。该系统以单片机为核心，配以 LCD 显示器、红外遥控器、步进电机模块，主要实现的功能有：

1）单片机实现对红外遥控器的解码。

2）根据按键选择，可控制步进电机的正反转、加速与减速。

3）单片机实时检测电机转速，并在 LCD 液晶显示屏上同步显示电机速度。

系统设计过程中，硬件电路极为简单，无需外围芯片，原理简单、工作稳定可靠、易于兼容。当发射器的类型不同，只需对中断处理程序的部分参数稍加改动即可，可以适用于多种红外遥控器信号的接收和解码，极大地节约了硬件实现的资源开销。将红外遥控器用在步进电机系统中，作为系统控制用红外遥控器，既操作灵活方便，又能提高系统抗干扰能力，在实际中收到了良好效果。设计的系统具有较强的灵活性和实用性，为新型遥控器材的研制做了有益的探索，具有一定的参考和借鉴作用。

## 21.2 系统总体设计

### 21.2.1 系统结构

红外遥控系统包括：Atmel 公司的 AT89S52 单片机最小系统、1602LCD 液晶显示电路、步进电机控制电路、6121 红外发射器电路、HS0038 红外接收头模块组成。系统的结构框图如图 21-1 所示。

本系统利用 AT89S52 单片机作为核心控制部分，当红外发射器发射经过编码和调制过的脉冲信号到一体化红外接收模块，然后红外接收模块解调出未解码的脉冲信号，然后把该信号送进单片机，单片机再进行解码，经过单片机解码后的脉冲信号再控制步进电机，一开始 LCD 显示 00 转/分，然后通过红外遥控器的按键不同，可以使步进电机做起动、停止、正转、反转、快转、慢转动作，在 LCD 上分别显示步进电机的速度。

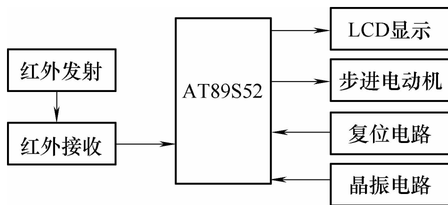



图 21-1 系统结构框图

### 21.2.2 红外遥控器工作原理

在很多单片机应用系统中，常常利用非电信号（如光信号、超声波信号等）传送控制信息和数据信息，以实现遥控或遥测的功能。红外通信具有控制简单、实施方便、传输可靠性高的特点，是一种较为常用的通信方式。因而，继彩电、录像机之后，在录音机、音响设备、空调机以及玩具等其他小型电器装置上也纷纷采用红外线遥控。工业设备中，在高压、辐射、有毒气体、粉尘等环境下，采用红外线遥控不仅完全可靠而且能有效地隔离电气干扰。

红外通信是利用950nm 近红外波段的红外线作为传递信息的媒体，即通信信道。发送

端采用脉位调制（PPM）方式，将二进制数字信号调制成某一频率的脉冲序列，并驱动红外发射管以光脉冲的形式发送出去；接收端将接收到的光脉冲信号转换成电信号，再经过放大、滤波等处理后送给解调电路进行解调，还原为二进制数字信号后输出。通用红外遥控系统由发射和接收两大部分组成。应用编/解码专用集成电路芯片来进行控制操作。红外线遥控器的系统框图如图 21-2 所示。发射部分包括键盘矩阵、编码调制、LED 发送器；红外接收部分包括光、电转换放大器、解调、解码电路。

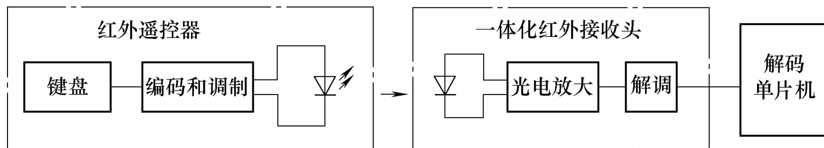



图 21-2　红外线遥控系统框图

遥控发射器专用芯片很多，本系统采用的红外遥控器是 NEC 的 UPD6121 编码（一般家庭用的 DVD、VCD、音响都使用这种编码方式）。当发射器按键按下后，即有遥控码发出，按键值不同，遥控编码也不同。这种遥控码具有以下特征：采用脉宽调制的串行码，以脉宽为 0. 565ms、间隔 0. 56ms、周期为 1. 125ms 的组合表示二进制的“0”；以脉宽为 0. 565ms、间隔 1. 685ms、周期为 2. 25ms 的组合表示二进制的“1”。其波形如图 21-3 所示。

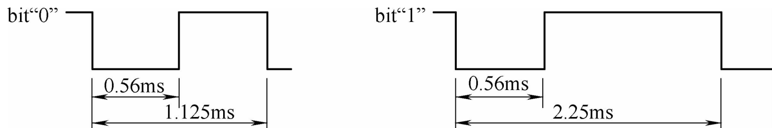



图 21-3　遥控码“0”和“1”编码

上述“0”和“1”组成的 32 位二进制码经 38kHz 的载频进行二次调制以提高发射效率，达到降低电源功耗的目的。然后再通过红外发射二极管产生红外线向空间发射，如图 21-4 所示。

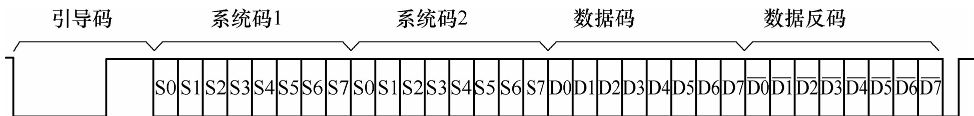



图 21-4　遥控信号编码波形图

UPD6121 产生的遥控编码是连续的 32 位二进制码组，其中前 16 位为用户识别码，能区别不同的电器设备，防止不同遥控码互相干扰。该芯片的用户识别码固定为十六进制 01H；后 16 位为 8 位操作码（功能码）及其反码。UPD6121 最多可有 128 种不同组合的编码。

遥控器在按键按下后，周期性地发出同一种 32 位二进制码，周期约为 108ms。一组码本身的持续时间随它包含的二进制“0”和“1”的个数不同而不同，大约在 45 ~ 63ms 之间，图 21-5 为遥控连发信号波形图。

当一个键按下超过 36ms，振荡器使芯片激活，将发射一组 108ms 的编码脉冲，这 108ms 发射代码由一个引导码（9ms），一个结果码（4. 5ms），低 8 位地址码（9 ~ 18ms），高 8 位地址码（9 ~ 18ms），8 位数据码（9 ~ 18ms）和这 8 位数据的反码（9 ~ 18ms）组成如图 21-6 和 21-7 所示。如果按键按下超过 108ms 仍未松开，接下来发射的代码（连发码）

将仅由起始码（9ms）和结束码（2. 25ms）组成。

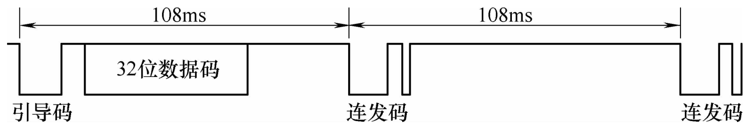



图 21-5 遥控连发信号波形

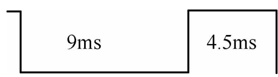



图 21-6 引导码

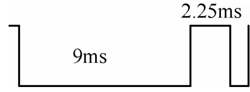



图 21-7 连发码

接收电路使用的是一种集红外线接收和放大于一体的一体化红外线接收器，它的优点是不需要复杂的调试和外壳屏蔽，不需要任何外接元件，就能完成从红外线接收到输出与 TTL 电平信号兼容的所有工作，而体积和普通的塑封晶体管大小一样，它适合于各红外线遥控和红外线数据传输。但在使用时要注意成品的红外接收头的载波频率。红外遥控最常用的载波频率为 38kHz，这是由于发射端使用的 455kHz 晶振决定的。接收器对外只有 3 个引脚：OUT、GND、VCC 与单片机连接非常方便，其内部结构框图如图 21-8 所示。

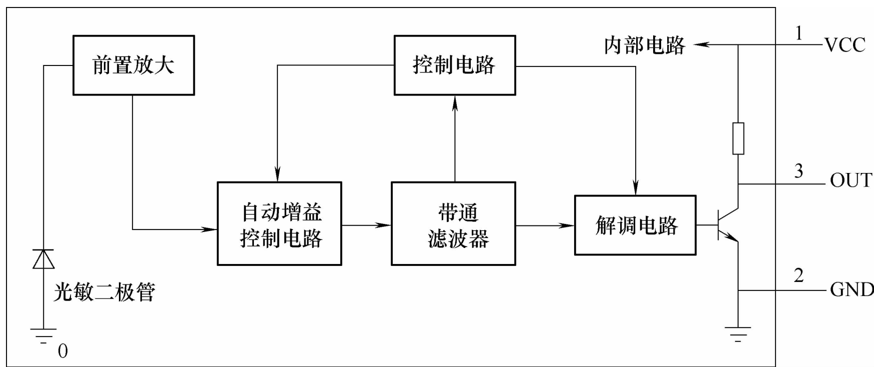



图 21-8 红外接收模块内部结构框图

接收模块的工作原理：首先，通过红外光敏元件将接收到的载波频率为 38kHz 的脉冲调制红外光信号转化为电信号，再由前置放大器和自动增益控制电路进行放大处理。然后，通过带通滤波器进行滤波，滤波后的信号由解调电路进行解调。最后，由输出级电路进行反向放大输出。为保证红外接收模块接收的准确性，要求发送端载波信号的频率应尽可能接近 38kHz，因此在设计脉冲振荡器时，要选用精密元件并保证电源电压稳定。

### 21. 2. 3 步进电动机工作原理

因步进电动机可精确实现所设定的角度和转数，具有良好的步进特性，最适合于数字控制，因此它在数控机床等设备中得到了广泛的应用。在工业被控设备对位移和角度控制要求较高的场所，步进电动机应用很多，而单片机芯片体积小、兼容性强、高速度、低价格、低工作电压、低功耗等特点，使单片机成为驱动步进电动机的最佳控制单元，所以基于单片机控制的步进电动机系统控制精度高、运行稳定，在控制领域有着广泛的应用。

本设计的功能是实现一个基于 AT89S52 单片机红外遥控控制的四相步进电动机系统，

由单片机将红外发射过来的编码进行解调后产生相应的驱动脉冲信号，步进电动机的驱动器收到驱动脉冲信号后，步进电动机将会按照设定的方向转动一个固定的角度，将电脉冲转化成角位移。电动机的转速由脉冲信号频率来控制决定，可以通过控制脉冲的个数来控制角位移量，从而达到调速的目的。一个完整的步进电动机控制系统包括控制器、驱动器、电动机三部分，其结构框图如图 21-9 所示。

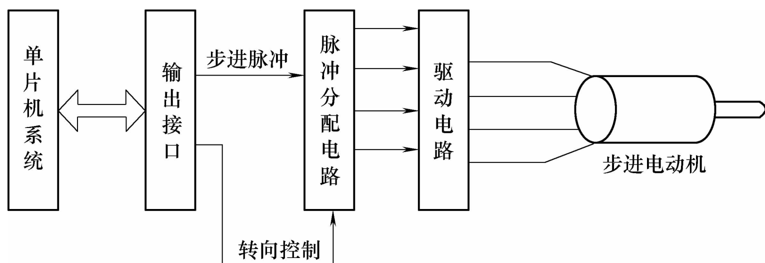



图 21-9　步进电动机模块框图

由图 21-9 可知，步进电动机控制系统中有两个重要电路：脉冲分配电路和驱动电路。脉冲分配电路有两个输入信号：步进脉冲和转相控制。脉冲分配电路在步进脉冲和转向控制信号的共同作用下产生四相激励信号，此激励信号经过驱动电路送至步进电动机，控制步进电动机向某一方向转动，此激励信号的频率决定了步进电机的转速。驱动电路的主要作用是实现功率放大，一般脉冲分配器输出的驱动能力是有限的，它不可能直接驱动步进电机，而要经过一级功率放大。

## 21.3　系统硬件电路设计

基于单片机的红外遥控系统，通过红外遥控器发射脉冲信号，红外线接收器对发射的脉冲信号进行接收，接收成功后，单片机对接收的脉冲信号进行处理，用来控制步进电动机的运行，电动机转动情况通过 LCD 可视化显示。系统硬件电路设计分成 4 个模块：MCU 核心控制模块、红外遥控模块、步进电机模块、1602LCD 显示模块。各个模块电路具体设计从四个部分介绍。

### 21.3.1　单片机最小系统设计

该模块在系统设计中是主要控制模块，电路如图 21-10 所示。该模块包括了 AT89S52 单片机、晶振电路和复位电路。本系统设计单片机采用 Atmel 公司的 AT89S52 芯片。AT89S52 提供以下标准功能：8KB Flash、256B RAM、32 位 I/O 口线、看门狗定时器、两个数据指针、3 个 16 位定时/计数器、1 个 6 向量 2 级中断结构、全双工串行口、片内晶振及时钟电路。另外，AT89S52 可降至 0Hz 静态逻辑操作，支持两种软件可选择节电模式：空闲模式下，CPU 停止工作，允许 RAM、定时/计数器、串口、中断继续工作；掉电保护方式下，RAM 内容被保存，振荡器被冻结，单片机一切工作停止，直到下一个中断或硬件复位为止。

在原理图 21-10 中，P1.1 ~ P1.4 作为 I/O 口控制步进电动机，P3.2 为红外遥控按键的中断输入口，P0 口作为 I/O 口数据到 LCD 显示器上显示，P2.0 ~ P2.2 口作为 I/O 口控制 LCD 显示。

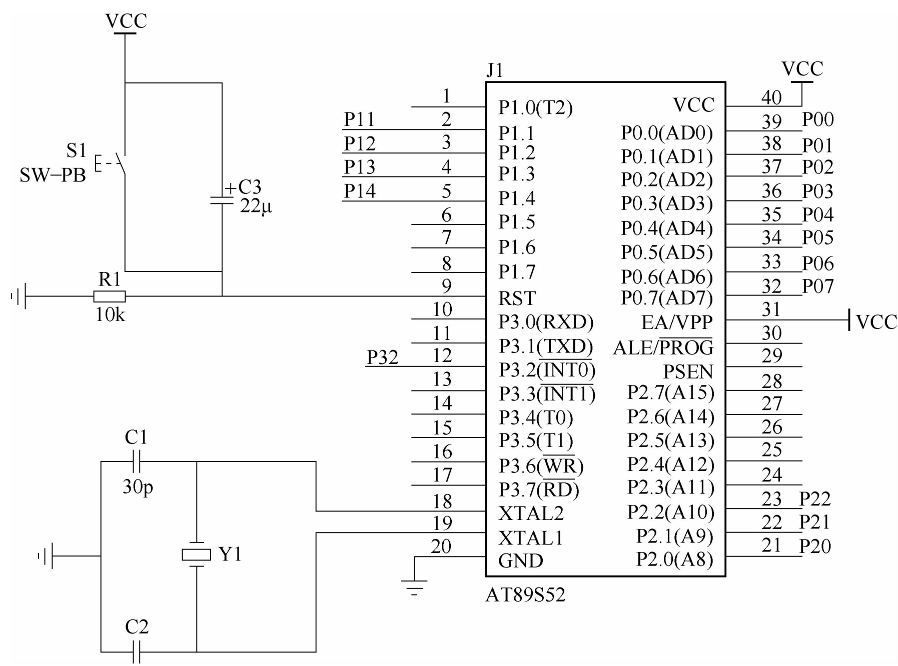



图 21-10 单片机最小系统电路

单片机要正常工作还需要有晶振电路和复位电路，因为没有晶振电路的话，也就没有时钟周期，无法执行程序代码，单片机也就无法工作，单片机的晶振电路是一种典型电路，分为内部时钟方式和外部时钟方式两种，本设计采用的是内部时钟方式，时钟的晶振频率选择 11.0592MHz，外接两个 30p 谐振电容。

而单片机的复位就和计算机的重起是一样的概念。任何单片机在工作之前都要有个复位的过程，复位对单片机来说，是程序还没有开始执行，是在做准备工作。一般的复位只需要 5ms 的时间。只要在单片机的 RST 引脚上加上高电平就完成了复位，按前面所说，时间不少于 5ms。为了达到这个要求，本设计在外部设计了复位电路，这种方法就是在复位电路中设计按键开关，通过按键开关触发复位电平，控制单片机的复位，当然 C3 电容可以起到一个上电自动复位的功能。

## 21.3.2 红外遥控器模块设计

项目选用集红外线接收和放大于一体的一体化红外线接收器 HS0038，它对外只有 3 个引脚：Out、GND、VCC 与单片机连接非常方便，如图 21-11 所示。其中 GND 接系统的地线（0V）；VCC 接系统的电源正极（5V）；脉冲信号输出接单片机的 P3.2（INT0）口。

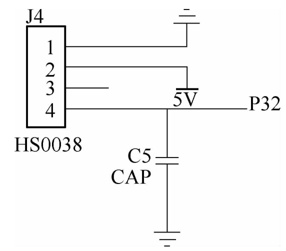



图 21-11 红外接收器电路图

## 21.3.3 步进电动机模块设计

步进电动机是纯粹的数字控制电动机，它将电脉冲信号转变成角位移，即给一个脉冲，步进电动机转一个角度，因此非常适合单片机控制。在非超载的情况下，电动机的转速、停止

的位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，电动机则转过一个步距角，同时步进电动机只有周期性的误差而无累积误差，精度高。

本设计选用驱动芯片 ULN2003，ULN2003 是由高压大电流达林顿晶体管阵列系列产品，具有电流增益高、工作电压高、温度范围宽、带负载能力强等特点，适应于各类高速大功率驱动系统。该模块的硬件电路如图 21-12 所示。

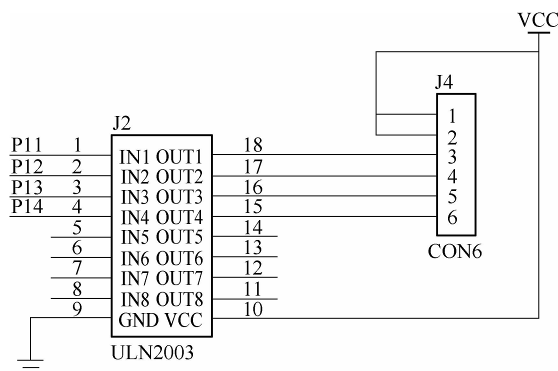



图 21-12　步进电动机驱动电路原理图

## 21.3.4　LCD 显示模块设计

液晶显示的原理是利用液晶的物理特性，通过电压对其显示区域进行控制，通电后可以显示出图形。液晶显示器具有厚度薄、适用于大规模集成电路直接驱动、易于实现全彩色显示的特点，目前已经被广泛应用在便携式计算机、数字摄像机、PDA 移动通信工具等众多领域。本文设计中的 1602LCD 采用的是字符型液晶显示模块，是一种专门用于显示字母、数字、符号等点阵式 LCD。

项目液晶显示选用 1602LCD，采用标准的 16 引脚（带背光）接口，其硬件电路如图 21-13 所示。图中 RP 为可调电阻，连接液晶显示偏压 VL，可调节 LCD 的亮度；P0.0 ~ P0.7 为 LCD 显示数据的输入口；P2.0 为数据/命令选择控制端 RS；P2.1 为读/写选择 R/W；BLA 和 BLK 为 LCD 的背光电源输入端。

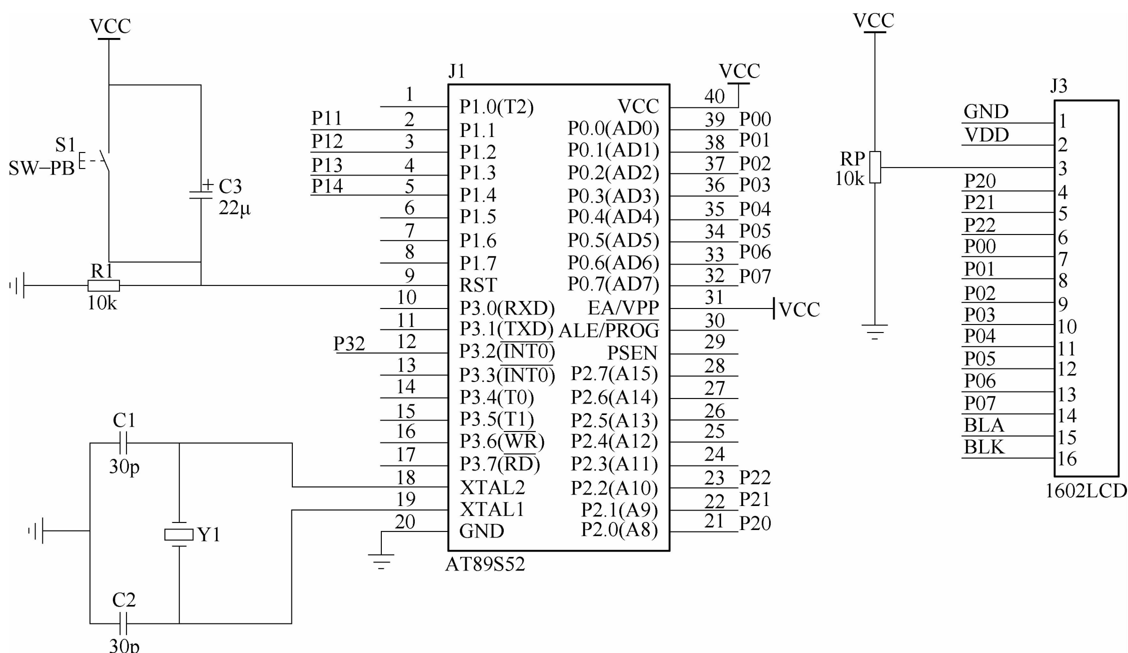



图 21-13　液晶显示电路

## 21.4 系统软件程序设计

本系统软件设计分为以下几个模块：主程序、红外遥控器解码程序、LCD 显示程序、步进电动机控制程序，下面对各个模块的工作原理流程设计具体介绍如下。

### 21.4.1 主程序设计

系统上电，1602LCD 显示 00 转/分，此时步进电动机处于停止状态。主程序设计首先初始化相关参数，如外部中断使用边沿触发、开外部中断等；红外接收器接收信号采用外部中断 INT0 中断方式，当有信号接收到时，系统进入中断服务程序，此时要关掉中断，目的是不让其他中断进来，影响红外的接收，也就是提高系统的抗干扰能力；红外中断服务程序实现键盘的处理，完成电动机控制功能键的处理：加速、减速、正转、反转、起动、停止等；LCD 显示程序完成对步进电动机转速的显示。该主程序的流程如图 21-14 所示。

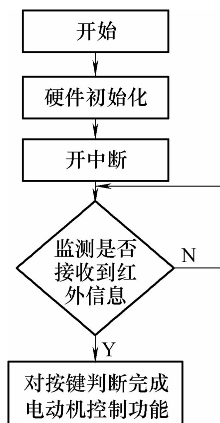



图 21-14 主程序流程图

主程序关键代码如下：

```
void main(){
    dy =2000;
    flag =2;
    sudu =0;
    IE =0x81;       //允许总中断中断,使能 INT0 外部中断
    TCON =0x01;   //触发方式为脉冲负边沿触发
    delay(1);
    IRIN =1;
    BEEP =1;
    lcd_show();
    while(1){
        if(flag ==0)zz();
        else if(flag ==1)fz();
    }
}
```

### 21.4.2 红外遥控器解码程序设计

红外遥控的解码就是从遥控器过来的信号，经过接收电路处理之后，将在输出端给出一个脉冲序列，而解码就是从这个脉冲序列中把信息恢复出来，不过注意很重要的一点：收到的信号是发射端信号的反相电平。

本设计对于 UPD6121 编码的解码，方法是获取电平的时间长度，然后按照给定参数识

别出信息。本设计采用的解码方法是采用硬件外部中断的方法，这种方法占用的硬件资源较少，但软件在中断程序里逗留的时间比较长，在一些时间敏感的场合不合适使用。软件解码程序的流程图如图 21-15 所示。

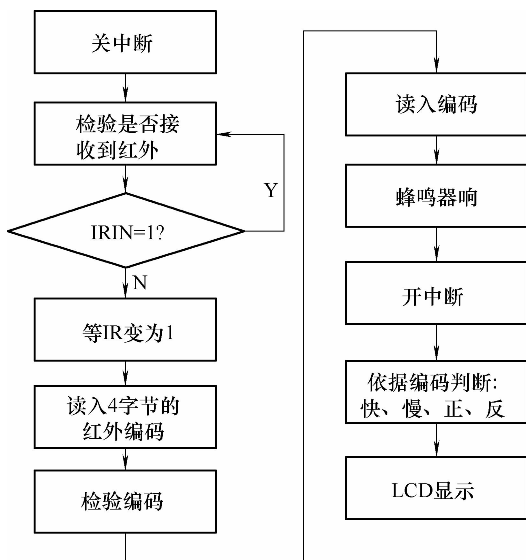



图 21-15　红外遥控器解码程序流程图

单片机 P3.2（IRIN）一检测到有红外数据发送过来，就进入中断服务程序。整个中断服务子程序主要分红外遥控器解码、蜂鸣器提示、键盘处理以及 LCD 显示等程序。红外遥控器解码程序分两步，第一步是读取并判断引导码是否正确，如果不是则直接返回并初始化检测参数；第二步是连续 4 次，按照每次 8 位读取 4 个字节的后续数据，其中，检测过程中对 0 和 1 的判断必须是在延时 0.56ms 之后，一般取 0.8ms 中间值，以减少因为程序执行而导致的测量时间长度上的误差。校验密码主要通过读入的 4 字节红外编码中，前两个是用户码和用户码的反码，后两个是键数据码和键数据码的反码，故只需将最后一个键数据反码取反后与键数据码相比较。若相等，则数据正确，将键值编码读入进行处理；若不相等，则数据读取错误，中断程序返回。红外遥控器发送过来的主要是以下五个键：正转、反正、加速、减速以及停止键。若是正转键按下，则电动机按照 A→B→C→D→A 方向依次通电；若是反转键按下，则电动机按照 A→D→C→B→A 方向依次通电；若是加速键按下，则减少单相通电的延时时间；若是减速键按下，则增加单相通电的延时时间；若是停止键按下，则电动机停止旋转。红外遥控接收的关键代码如下：

```
void IR_IN( ) interrupt 0 using 0
{
    unsigned char i,j,k,N =0,sel;
    EA =0;
    do{
        for( i =0 ;i <4 ;i ++ ) {
            if( IRIN ==0) break;
            if( i ==3) { EA =1 ;return ;}
        }
        delay(20);
    } while( IRIN ==1);//确认 IR 信号出现
while( ! IRIN) {delay(1);}//等 IR 变为高电平,跳过 9ms 的前导低电平信号
for( j =0;j <4;j ++ )//收集四组数据
{
    for( k =0;k <8;k ++ )//每组数据有 8 位
```

```
        {
            while(IRIN){delay(1);}    //等IR变为低电平,跳过4.5ms的前导高电平信号
            while(! IRIN){delay(1);}    //等IR变为高电平
            while(IRIN)               //计算IR高电平时长
            {
                delay(1);
                N++;
                if(N>=30){EA=1;return;}//0.14ms计数过长自动离开
            }//高电平计数完毕
            IRCOM[j]=IRCOM[j]>>1;//数据最高位补“0”
            if(N>=8){IRCOM[j]=IRCOM[j]|0x80;}//数据最高位补“1”
            N=0;
        }
    }
    if(IRCOM[2]!=~IRCOM[3]){EA=1;return;}//校验数据是否正确,不正确则返回
    IRCOM[5]=IRCOM[2] & 0x0F;
    IRCOM[6]=IRCOM[2] & 0xF0;
    IRCOM[6]=IRCOM[6]>>4;
    beep();
    EA=1;
    lcd_show();
}
```

### 21.4.3 LCD显示程序

1602LCD主要是用来显示步进电动机的转速，因为项目设计所用的电动机额定转速在100r/s之内，所以只要取步进电动机速度的十位和个位，送到LCD上就可以了。LCD显示的具体流程如图21-16所示。

液晶显示模块是一个慢显示器件，所以在执行每条指令之前一定要确认模块的忙标志为低电平（即空闲状态），否则此指令失效。要显示字符时要先输入显示字符地址，也就是告诉模块在哪里显示字符。程序在开始时对液晶模块功能进行了初始化设置，约定了显示格式。注意显示字符时光标是自动右移的，无需人工干预，每次输入指令都先调用判断液晶模块是否忙的子程序delay，然后输入显示的地址，最后是输入要显示的字符代码。LCD的读写操作、屏幕和光标的操作都是通过指令编程来实现的。

LCD初始化关键代码如下：

```
lcd_init(){                    //LCD初始化设定
lcd_wcmd(0x38);                //功能设置
delay3(1);
lcd_wcmd(0x0c);                //开显示
delay3(1);
```

```
    lcd_wcmd(0x06);                //输入模式设置
    delay3(1);
    lcd_wcmd(0x01);                //清除 LCD 的显示内容
    delay3(1);
}
```

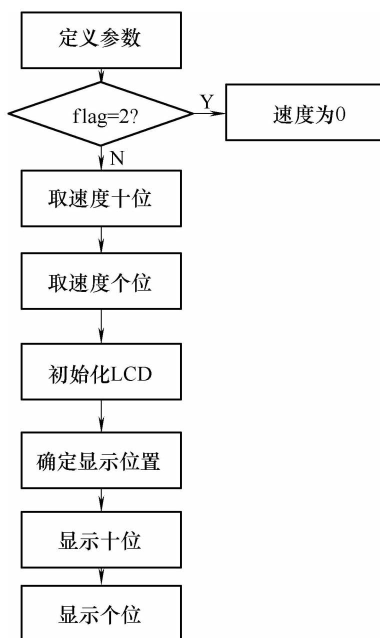



图 21-16　LCD 显示程序流程图

### 21.4.4　步进电动机控制程序

步进电动机的位置和速度与导电次数（脉冲数）和频率成一一对应关系，而方向由导电顺序决定。四相步进电动机按照通电顺序的不同，可分为单四拍、双四拍、八拍三种工作方式。本程序设计时采用单四拍工作方式，即在每一瞬间只有一个线圈导通。该工作方式具有如下特点：消耗电力小、精确度良好、但转矩小、振动较大，每送一励磁信号可走 5.625°。若以 A→B→C→D→A 相励磁顺序控制步进电动机则正转，若以 A→B→C→D→A 相励磁顺序控制步进电动机则反转。

根据红外接收头对接收的信号进行解码，对电动机的快、慢、正和反进行设定，其部分关键代码如下：

```
    sel = IRCOM[6] * 10 + IRCOM[5];
    if(sel == 36){dy = dy/1.5;flag = 0;}
    else if(sel == 37){dy = dy * 1.5;flag = 0; }
    else if(sel == 38){ flag = 1;}
    else if(sel == 39){ flag = 0;}
    else if(sel == 23){ flag = 2;}
```

电动机正转关键代码如下：

```
void zz(){
P1 =0xf6;
delay2(dy);
P1 =0xf7;
delay2(dy);
P1 =0xf3;
delay2(dy);
P1 =0xfb;
delay2(dy);
P1 =0xf9;
delay2(dy);
P1 =0xfd;
delay2(dy);
P1 =0xfc;
delay2(dy);
P1 =0xfe;
delay2(dy);
}
```

电动机反转关键代码如下：

```
void fz(){
P1 =0xfe;
delay2(dy);
P1 =0xfc;
delay2(dy);
P1 =0xfd;
delay2(dy);
P1 =0xf9;
delay2(dy);
P1 =0xfb;
delay2(dy);
P1 =0xf3;
delay2(dy);
P1 =0xf7;
delay2(dy);
P1 =0xf6;
delay2(dy);
}
```

## 21.5　系统调试总结

调试中遇到的问题主要是在红外中断程序设计时，需要配合红外遥控器、LCD 和步进电动机的转动，将 LCD 显示和按键操作放入中断服务子程序中，考虑信号不被干扰，所以首先设置了进入中断后关中断，然后要检测是否是红外信号，通过不断地对软件和硬件联合调试确保软件设计正确性。

## 附件：系统设计的电路原理图

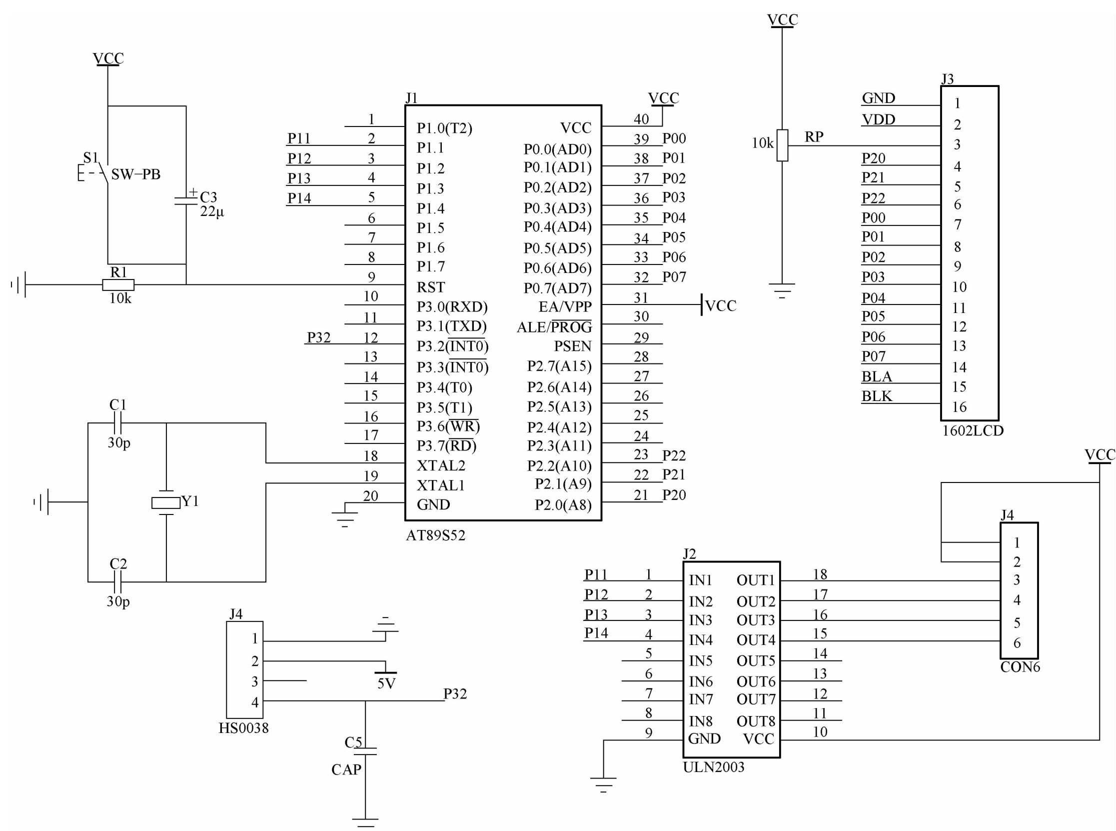



图 21-17　系统设计电路原理图

# 第 22 章　智能小车自动寻迹系统设计

## 22.1　项目背景和研究内容

### 22.1.1　项目背景

随着计算机技术、信息技术、控制技术的快速发展，工业的生产和管理进入了自动化、信息化和智能化时代，因此智能化已经成为时代发展的必然需要。在柔性自动化生产线、智能仓储管理及物流配送等领域，当生产现场环境恶劣时，人工不能完成的任务如物料运输和装卸等，都可采用智能寻迹小车去完成相应的任务。基于生产现场和日常生活的实际需要，研究和开发智能小车寻迹系统具有十分重要的意义。因此该项目的研究可以应用于工厂自动化、机车前照灯自动寻迹、仓库管理、智能玩具和民用服务等领域，可提高劳动生产效率，改善劳动环境。

对于此项研究，国内有很多高校都在研究。其中浙江省的电子设计竞赛就出现了设计自动寻迹小车。当然最有名的是飞思卡尔智能车大赛，都采用光电传感技术来实现小车的自动寻迹。因此基于自动寻迹的智能小车系统设计的研究与开发，不仅具有重要的理论意义，而且具有重要的工程应用价值，在今后的自动化控制中将起到重要的作用。

### 22.1.2　研究内容

项目设计任务是设计并制作一套用电池供电的智能小车系统，包括一台能沿着黑色引导线自主行驶的小车和一个电子公交站，小车行驶线路为任意曲线，其行驶的距离通过无线传输模块传输到站台上进行显示。本系统设计以单片机和 CPLD 作为主要的控制芯片，它们具有编程灵活、集成度高、设计开发周期短、适用范围宽、开发工具先进、设计制造成本低的优点。利用 CPLD 产生 PWM 波来精确的控制小车两轮的速度。利用 RPR220 光电传感器来实现小车轨迹的自动寻迹，其工作原理是红外发射管向某一方向发射红外光，遇到黑色的轨迹线，红外光被吸收，从而接收管接收不到光线而截止；遇到白色的轨迹线，红外光被反射，从而接收管接收光线而导通，根据接收管的导通与截止输出“0”或“1”信号传送给 CPLD 进行处理并给出相应的直行、左转弯、右转弯等命令，从而使得小车沿着拟定的任意黑色轨迹线前行。本系统采用单片机 AT89S52 来实现小车行驶距离的计算、小车与站台间的无线通信以及数码管距离的显示。

### 22.1.3　系统设计技术

**1. 光电传感技术**

光电传感器是采用光电元件作为检测元件的传感器。它首先把被测量的变化转换成光信号的变化，然后借助光电元件进一步将光信号转换成电信号。光电传感器一般由光源、光学

通路和光电元件三部分组成。光电检测方法具有精度高、反应快、非接触等优点，而且可测参数多、传感器的结构简单、形式灵活多样。因此，光电式传感器在检测和控制中应用非常广泛。光电传感器具有以下特点：

（1）检测距离长

如果在对射型中保留 10m 以上的检测距离，便能实现其他检测手段（磁性、超声波等）无法达到长距离检测。

（2）对检测物体的限制少

由于以检测物体引起的遮光和反射为检测原理，所以不像接近传感器等将检测物体限定在金属，它可对玻璃、塑料、木材、液体等几乎所有物体进行检测。

（3）响应时间短

光本身为高速，并且传感器的电路都由电子零件构成，所以不包含机械性工作时间，响应时间非常短。

（4）分辨率高

能通过高级设计技术使投光光束集中在小光点，或通过构成特殊的受光光学系统，来实现高分辨率。也可进行微小物体的检测和高精度的位置检测。

（5）可实现非接触的检测

可以无须机械性地接触检测物体实现检测，因此不会对检测物体和传感器造成损伤。因此，传感器能长期使用。

（6）可实现颜色判别

通过检测物体形成光的反射率和吸收率，根据被投光的光线波长和检测物体的颜色组合而有所差异。利用这种性质，可对检测物体的颜色进行检测。

**2. 无线通信技术**

这些年来，全球通信技术发生了翻天覆地的变化，尤其是近两三年来，无线通信技术的发展速度与应用领域已经超过了固定通信技术，呈现出如火如荼的发展态势。其中最具代表性的有宽带无线接入、蜂窝移动通信，也包括集群通信、卫星通信以及手机视频业务与技术。与其他方式相比，用无线数传模块建立专用无线数据传输方式具有如下优点，下面介绍一下用无线数传模块建立专用无线数据传输方式相比于有线通信的优点：

（1）成本廉价

有线通信方式的建立必须架设电缆或挖掘电缆沟，因此需要大量的人力和物力。而用无线数传电台建立专用无线数据传输方式则无需架设电缆或挖掘电缆沟，只需要在每个终端连接无线数传电台和架设适当高度的天线就可以了。相比之下用无线数传模块建立专用无线数据传输方式，节省了人力和物力，投资是相当节省的。当然在一些近距离的数据通信系统中，无线的通信方式并不比有线的方式成本低，但是有时候实际的现场环境难以布线，客户根据现场环境的需要还是会选用无线的方式来实现通信。

（2）建设工程周期短

当要把相距数公里到数十公里距离的远程站点相互连接通信的时候，采用有线的方式，必须架设长距离的电缆或者挖掘漫长的电缆沟，这个工程周期可能就需要数个月的时间，而用数传模块建立专用无线数据传输的方式，只需要架设适当高度的天线，工程周期只需要几天或者几周就可以，相比之下，无线的方式可以迅速组建起通信链路，工程周期大大缩短。

（3）适应性好

有线通信的局限性太大，在遇到一些特殊的应用环境，如遇到山地、湖泊、林区等特殊的地理环境或是移动物体等布线比较困难的应用环境的时候，将对有线网络的布线工程有着极强的制约力，而用无线数传模块建立专用无线数据传输方式将不受这些限制，所以说用无线数传模块建立专用无线数据传输方式将比有线通信有更好、更广泛的适应性，几乎不受地理环境限制。

（4）设备维护上更容易实现

有线通信链路的维护需沿线路检查，出现故障时，一般很难及时找出故障点，而采用无线数传模块建立专用无线数据传输方式只需维护数传模块，出现故障时则能快速找出原因，恢复线路正常运行。

本设计则采用了 PT2272 无线发射和 PT2262 无线接收模块，通过对其地址进行编码，实现两块模块的连接。对其编码的方式是分别对 PT2262 和 PT2272 的地址引脚悬空，实现地址的匹配，然后分别进行编码。设计采用了数据传输的基本方法，即先发送起始位和数据位，最后发送结束位。起始位使用 0x0C，结束位采用 0x0E，从而实现数据的传输。

## 22.2 系统电路设计

### 22.2.1 系统工作原理

本系统设计采用左右两个传感器来实现小车行进路径的探测，并将相应的“0”或“1”信号反馈给主控制芯片 CPLD 和单片机，CPLD 根据传感器的 0/1 信号产生相应的 PWM 波形来控制小车两轮速度，其中直行时产生两轮速度相同（即两路 PWM 的占空比相同），左转弯时右轮速度大于左轮速度（即右轮的占空比比左轮的大），右转弯时左轮速度大于右轮速度（即左轮的占空比比右轮的大）。单片机 AT89S52 主要任务是实现小车行驶距离的计算，并通过无线发射模块将数据传送给电子公交站显示。系统框图如图 22-1 所示。

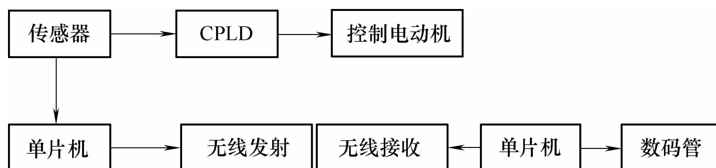



图 22-1 系统框图

### 22.2.2 系统硬件电路设计

硬件电路选用 Alter 公司的 EPM7128SLC84-10 芯片作为 CPLD 核心控制器，实现小车直行、左转弯与右转弯的控制；选用 Atmel 公司的 AT89S52 作为控制芯片，实现小车速度的显示以及无线通信；选用两个 RPR220 作为传感器模块，来实现小车行驶路线的寻迹；选用两片 L298N 作为直流电机的驱动芯片；选用 PT2272 无线发射和 PT2262 接收模块进行数据的传送；选用 74LS07 作为锁存器和 6 个八段共阴数码管作为显示模块。

**1. CPLD 模块设计**

CPLD 是一种用户根据各自需要而自行构造逻辑功能的数字集成电路。其基本设计方法是借助集成开发软件平台，用原理图、硬件描述语言等方法，生成相应的目标文件，通过下载电缆（“在线系统” 编程）将代码传送到目标芯片中，实现设计的数字系统。而本设计采用的 EPM7128SLC84-10 是 Alter 公司生产的 CPLD 芯片。其具体 I/O 口的分配为 15、16、17、18 4 个作为输出脚，分别与电动机的驱动芯片 ABCD 相连，作为直流电机左右轮的驱动信号。83 针脚与时钟芯片产生的 clk 时钟引脚相连，从而为控制芯片提供时钟，此 I/O 口为输入端。75、76 引脚与传感器输出相连，作为传感器信号的输入引脚，其电路图如图 22-2 所示。系统选用 12MHz 晶振作为时钟电路，给系统工作提供基本的时钟基准信号；利用 JTAG 口进行数据的下载。

图 22-2　CPLD 控制电路

**2. AT89S52 单片机模块设计**

AT89S52 是一种低功耗、高性能、由 CMOS 工艺制成的 8 位控制器，具有 8K 字节可编程 Flash 存储器，由于 Atmel 公司采用高密度非易失性存储制造工艺，使得 AT89S52 与 89C51 单片机的指令和引脚完全兼容，用过 89C51 的读者可以快速掌握 89S52 的使用方法。其中片上 Flash 允许程序存储器在线编程，适用于常规编程器，促使 AT89S52 为众多嵌入式控制应用系统提供灵活、有效的解决方案。

本系统设计中采用两片单片机，实现小车距离的计算与站台显示，资源分配具体如下：小车控制单片机芯片采用 P1.1，P1.2 连接两个传感器，进行路程计算；P0 口作为无线发射

模块的输出口，实现数据的发送，如图22-3所示。而站台控制单片机芯片采用P0口输出控制数码管的段码信号；P2口输出控制数码管的位码选通信号；P1口作为无线接收模块实现数据接收，电路图如图22-4所示。

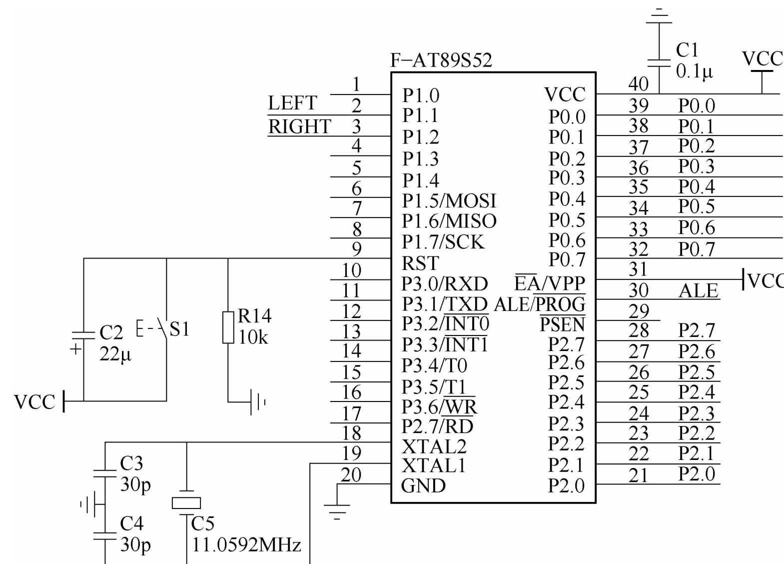



图22-3　小车单片机电路原理图

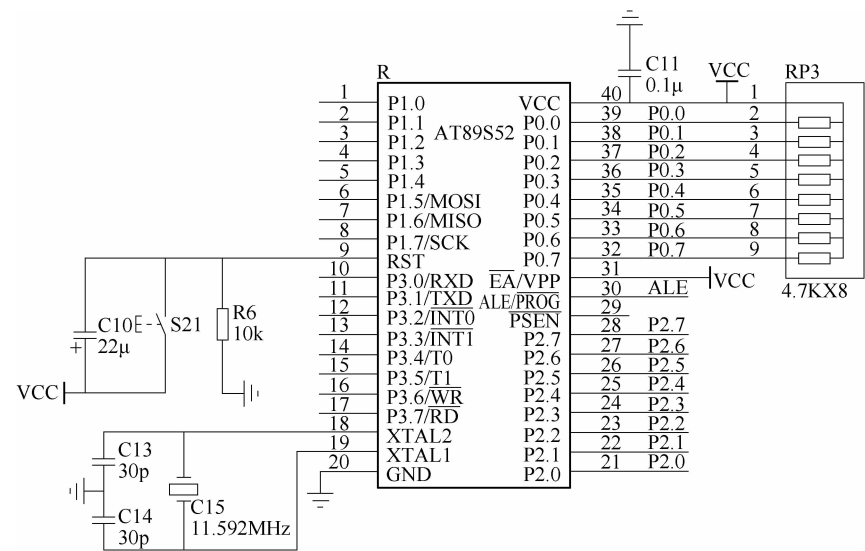



图22-4　站台控制单片机电路原理图

系统采用11.0592MHz晶振作为时钟电路，给系统工作提供基本的时钟基准信号；采用手动加上电复位作为系统的复位电路，使得电路在发生异常时可随时复位回到初始状态。

**3. 传感器模块设计**

该模块主要采用的芯片是RPR220芯片，该芯片具有如下特点：塑料透镜可以提高灵敏度，内置可见光过滤器能减小离散光的影响，体积小、结构紧凑；当发光二极管发出的光遇到白纸反射回来时，晶体管导通输出低电平，否则晶体管截止输出高电平。因此光电管调理

电路简单，工作性能稳定，传感器测量距离为 6mm，安装在小车底部与地面相距 5mm。传感器设计电路如图 22-5 所示。

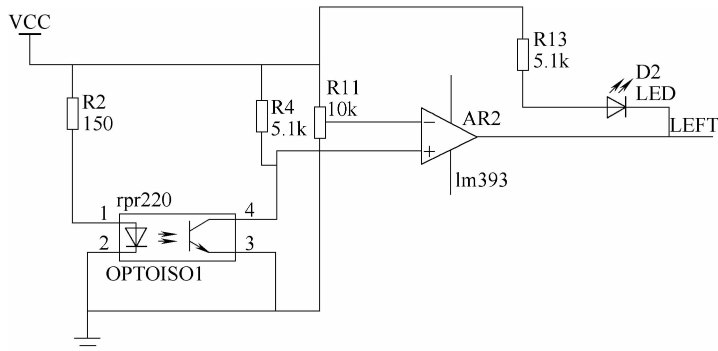



图 22-5　传感器检测与调理电路图

电路中，R2 起限流电阻的作用，R4 起上拉电阻的作用，使得 RPR220 在检测到黑色轨迹晶体管截止时能保证输出高电平。由于 RPR220 的输出电压（IC）与光照（输入电流 IF）呈线性关系，为了提高系统的抗干扰能力和系统的稳定性，因此在 RPR220 输出端加上 LM393 电压比较器，输出稳定的“0”或“1”信号。光电对管检测可调电阻 R11 可以调节比较器的门限电压，经示波器观察，输出波形为稳定的高低电平。由于 LM393 电压比较器是集电极开路输出的，所以必须加上拉电阻 R13，才能输出高电平。同时为了方便调试，接一个 LED 作为 0/1 信号的指示灯。当 RPR 检测到黑色轨迹线时，电压比较器电路输出低电平，此时 LED 发光，否则 LED 不亮。

**4. 电动机驱动模块设计**

本设计采用 L298N 作为电动机驱动芯片，L298 是 SGS 公司的产品，比较常见的是 15 脚 Multiwatt 封装的 L298N，内部同样包含 4 通道逻辑驱动电路，可以方便地驱动两个直流电机或一个两相步进电动机。由于 L298N 的逻辑电源采用 5V 供电，故其 5、7、10、12 4 个引脚可直接连接到 CPLD 上，通过对 CPLD 的编程就可以实现两个直流电动机的 PWM 调速以及正反转等功能，L298N 控制逻辑真值表见表 22-1。

**表 22-1　L298N 控制逻辑真值表**

| 输　入 | | 状　态 |
|---|---|---|
| ENABLE B = H | INPUT 3 = H，INPUT 4 = L | 电动机正转 |
| | INPUT 3 = L，INPUT 4 = H | 电动机反转 |
| | INPUT 3 = L，INPUT 4 = L | 电动机停止 |

L298N 芯片可以驱动两个两相电动机，也可以驱动一个四相电动机，输出电压最高可达 50V，可以直接通过电源来调节输出电压，驱动电路如图 22-6 所示。

**5. 无线通信模块设计**

PT2262/PT2272 是中国台湾普城公司生产的一种 CMOS 工艺制造的低功耗、低价位通用编解码电路。PT2262/PT2272 可有 12 位（A0 ~ A1）三态地址端引脚（悬空，接高电平，接低电平），任意组合可提供 531441 个地址码，PT2262 最多可有 6 位（D0 ~ D5）数据端引脚，而设定的地址码和数据码是从芯片 17 脚串行输出。

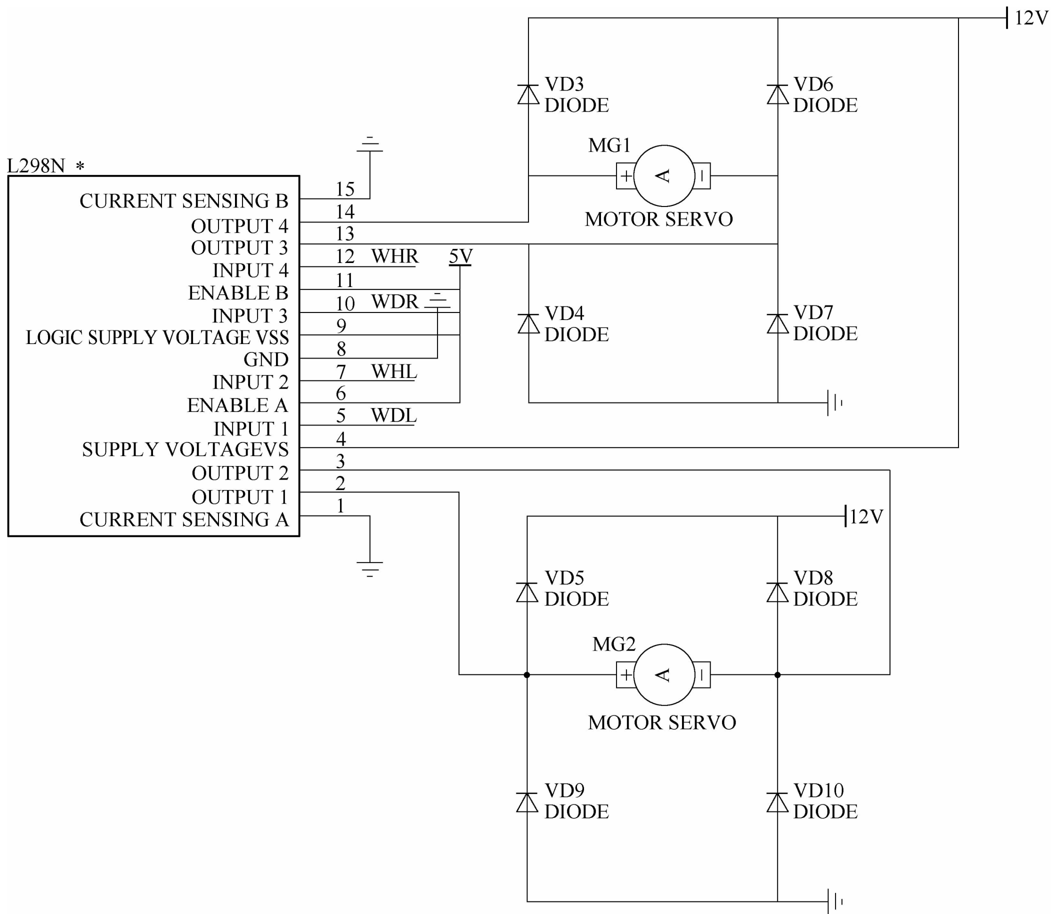



图 22-6 电动机驱动电路原理图

（1） PT2262 芯片

该编码芯片 PT2262 发出的编码信号由地址码、数据码、同步码三者组成一个完整的码字，当解码芯片 PT2272 接收到信号后，其地址码经过两次比较核对后，VT 脚才输出高电平，与此同时相应的数据脚也输出高电平，如果发送端一直进行数据发送，那么编码芯片也会连续发射。当发射机没有发送时，PT2262 不接通电源，其 17 脚为低电平，所以 315MHz 的高频发射电路不工作。当有数据发送时，PT2262 就进行工作，其第 17 脚就输出经调制的串行数据信号，当 17 脚为高电平期间 315MHz 的高频发射电路起振并发射等幅高频信号，当 17 脚为低电平期间，315MHz 的高频发射电路就停止振荡，不工作。所以高频发射电路完全取决于 PT2262 的 17 脚输出的数字信号，从而对高频电路完成幅度键控（ASK 调制）相当于调制度为 100% 的调幅。PT2262 的引脚说明见表 22-2。

本次系统设计中 A0 ~ A7 悬空，D0 ~ D3 作为数据的输入端，与单片机相连，并由外围电路将数据发射出去。由于考虑到外围电路要采用到高频信号，因此采用的是集成电路模块。

（2）PT2272 芯片

PT2272 解码芯片有不同的后缀，表示不同的功能，其中有 L4/M4/L6/M6 之分，其中 L 表示锁存输出，数据只要成功接收信号就会一直保持对应的电平状态，直到下次传送数据发

生变化时改变。M 表示非锁存输出，数据脚输出的电平是瞬时的，并且这和发射端是否发射相关，可以用于类似点动控制的方式。而后缀的 6 和 4 表示有几路并行的控制通道，即采取的几位数据传输。当采用 4 路并行数据时（PT2272-M4），对应的地址编码应该是 8 位，如果采用 6 路的并行数据时（PT2272-M6），对应的地址编码应该是 6 位。其引脚说明见表 22-3。

**表 22-2　PT2262 引脚说明**

| 名称 | 引脚 | 说　　明 |
|---|---|---|
| A0 ~ A11 | 1 ~ 8、10 ~ 13 | 地址引脚，用于进行地址编码，可置为“0”，“1”，“f”（悬空） |
| D0 ~ D5 | 7 ~ 8、10 ~ 13 | 数据输入端，有一个为“1”即有编码发出，内部下拉 |
| VCC | 18 | 电源正端（ + ） |
| VSS | 9 | 电源负端（ - ） |
| TE | 14 | 编码启动端，用于多数据的编码发射，低电平有效 |
| OSC1 | 16 | 振荡电阻输入端，与 OSC2 所接电阻决定振荡频率 |
| OSC2 | 15 | 振荡电阻振荡器输出端 |
| Dout | 17 | 编码输出端（正常时为低电平） |

**表 22-3　PT2272 引脚说明**

| 名称 | 引脚 | 说　　明 |
|---|---|---|
| A0 ~ A11 | 1 ~ 8、10 ~ 13 | 地址引脚，用于进行地址编码，可置为“0”，“1”，“f”（悬空），必须与 2262 一致，否则不解码 |
| D0 ~ D5 | 7 ~ 8、10 ~ 13 | 地址或数据引脚，当做为数据引脚时，只有在地址码与 2262 一致，数据引脚才能输出与 2262 数据端对应的高电平，否则输出为低电平，锁存型只有在接收到下一数据才能转换 |
| VCC | 18 | 电源正端（ + ） |
| VSS | 9 | 电源负端（ - ） |
| DIN | 14 | 数据信号输入端，来自接收模块输出端 |
| OSC1 | 16 | 振荡电阻输入端，与 OSC2 所接电阻决定振荡频率 |
| OSC2 | 15 | 振荡电阻振荡器输出端 |
| VT | 17 | 解码有效确认输出端（常低）解码有效变成高电平（瞬态） |

PT2272 与 PT2262 芯片的编码一致才能传输，所以本文中 PT2272 的 A0 ~ A7 也悬空，使得其编码方式一样，D0 ~ D3 作为数据的输入端，与单片机相连，为了解决单片机 P0 的复用功能，因此连接了 74LS373 芯片作为数据锁存器，对 P0 口进行扩展使用。

**6. 数码管显示电路设计**

数码管是由 8 段发光二极管组成，成一个日字形，它们可以是共阴极的，也可以是共阳极的。每个数码管都是按 a、b、c、d、e、f、g 七个笔画和一个小数点 DP 的顺序排列的，这 8 个对应二极管如果阳极连在一起就称共阳极，阴极连在一起称共阴极。数码管可以通过硬件解码电路得到相应的段码，也可以通过软件查询段码表得到相应数字的段码。本系统设计中采用动态扫描方式，P0 控制数码管的段码，而 P2 口控制数码管的位码，6 个数码管显示电路原理图如图 22-7 所示。

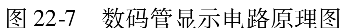


图 22-7　数码管显示电路原理图

## 22.3　系统软件设计

系统设计主要完成的任务有小车的自动寻迹、数据的无线传输以及距离的显示，因此软件设计分为以下几个模块：主程序、无线发射和接收程序设计、数码管动态显示程序设计，下面对各个模块的工作原理以及工作流程具体介绍如下。

### 22.3.1　主程序设计

系统上电，主程序开始检测传感器的检测信号，left1，Right1 分别为左右两个传感器的输入信号，（rout1、rout2）分别为控制右轮的高低电平输出，（lout1，lout2）分别为控制左轮的高低电平输出。系统主程序流程如图 22-8 所示。



图 22-8　主程序流程图

直流电动机的工作的原理：电动机速度的快慢不仅跟电压有关还跟其占空比有关。本设计采用定电压，不同占空比来控制。其核心程序如下：

```
process(clk)
    begin
        if(clk'event and clk ='1')then
            if(rcount =6000)then
```

```
                rcount <= 0;
            else
                rcount <= rcount + 1;   --产生 6000 个时钟信号
if(left1 ='0' and right1 ='0')then
            if(rcount <5000) then      --对 6000 个时钟信号进行分频,前 5000 为信号“1”
                rout1 <='1';            --左右轮为正常速度
                lout1 <='1';
            else
                rout1 <='0';
                lout1 <='0';
                end if;
elsif(left1 ='0' and right1 ='1')   then
            if(rcount <4000) then       --右轮为减速行驶
                  rout1 <='1';
                else
                  rout1 <='0';
                  end if;
            if(rcount <5000) then      --左轮正常行驶
                  lout1 <='1';
                else
                  lout1 <='0';
                  end if;
```

该程序首先产生 6000 个时钟，并对 6000 个时钟进行分频来控制占空比。小车正常运行给其 5000/6000 的占空比。而对与转弯时的速度，给予 4000/6000 占空比。当转小车转弯时，两电动机实现不同的速度就可实现转弯。在 Quartus II 软件对代码进行编译综合后，其仿真时序波形如图 22-9 所示。

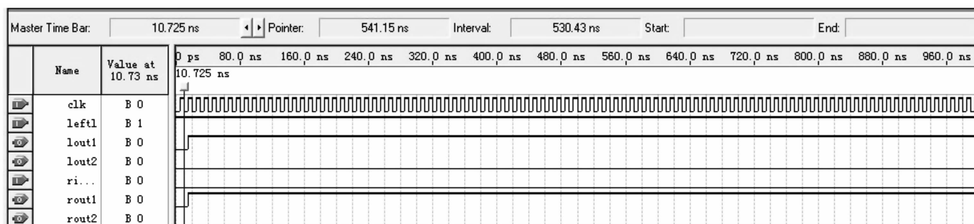



图 22-9　仿真时序波形图

### 22.3.2　无线发射程序设计

本系统中无线模块的接收和发送是比较难解决的问题，采用类似于 UTP 的传输方式进行传输。同时，由于该发射接收模块只有 4 个数据端口，因此如果需要传送数据就要进行数据的编码，从 1 ~9 数字的编码，传送数据就要一位一位的传送。为了保证数据传输的正确性，本系统设计采用串行通信传输方法，即先发送起始位和数据位，最后发送结束位。起始

位使用0x0C，结束位采用0x0E，从而实现数据的传输。具体的操作如下：单片机对传感器信号进行处理，判断小车处于的状态，并对小车转速进行计算，将数据发送给站台。传送方式采用类似的UTP传输。发送接收信号，当站台单片机接收到接收信号，便开启数据的接收。直到站牌单片机收到结束信号，便进行数据的运算，得到小车离站的距离，通过站牌数码管显示。其数据传输的核心代码如下：

```
//数据传输时的时间处理
void int_t0(void) interrupt 1 using0      //采用计数器0的工作方式为1
{
    TH0 =0x3c;   /*发射*/
    TL0 =0x0b0;
        fg ++;
        if(fg ==20)                    //1s传输一次
         {if(k1 ==0&&k2 ==0)      /K1为左传感器K2为右传感器      /
         {
         count ++;                     /*计算时间*/
         }
           send =1;
           fg =0;
         }
}
```

该程序主要完成数据的定时发送。

### 22.3.3 数码管动态显示程序设计

本系统中采用P0口输出控制数码管的段码信号；P2口输出控制数码管的位选通信号。当无线模块传来数据后进行数据处理后，利用动态扫描的方式进行显示数据，主要处理程序如下：

```
do                       /* y为传入数据*/
{
c =y%10;                 /* c为保存每一位的数据*/
y =y/10;
P0 =disptab[c];          /*显示段码*/
P2 =ctltab[a ++];        /*显示位码*/
for(i =0;i <255;i ++);
}
while(y! =0);
```

利用以上程序便可以动态显示数据。

## 22.4 系统调试

系统设计主要采用QUARTUSⅡ和Keil软件进行程序开发，用VHDL语言编写程序来完

成传感器信号的检测以及直流电机速度和方向的控制设计，用 C 语言来实现无线通信模块的接收和发送以及数码管的显示。利用 PROTEL 99SE 进行系统电路原理图设计，然后使用面包板进行实际焊接。为了防止焊接中的虚焊情况，焊接过程用万用表来进行测试。

调试中遇到的主要问题有

①如何实现小车的自动寻迹；

②如何安排传感器位置；

③如何实现正确的无线传输数据。

对于第一个问题，其解决方法是利用 CPLD 来输出两路 PWM 波来控制小车两轮的速度，以保证小车能按照轨道直行、左转弯或右转弯。对于小车速度的控制（即 PWM 的占空比）是经过多次实际调试后，才选择了一个比较合理的速度。

对于第二个问题中传感器位置的确定，因为传感器测量的有效距离为 6mm 左右。为了保证测量的正确性，通过支架来固定，并且保证左右传感器的水平高度一样。

对于第三个问题中主要是无线传输模块无法进行数据传输。经过调试发现是传输的时序控制不对，最后编程确定有效的时序使得双方能进行正确的数据传输。另外，焊接电路时要格外的小心，以避免虚焊与短焊。

## 附件：系统设计的电路原理图

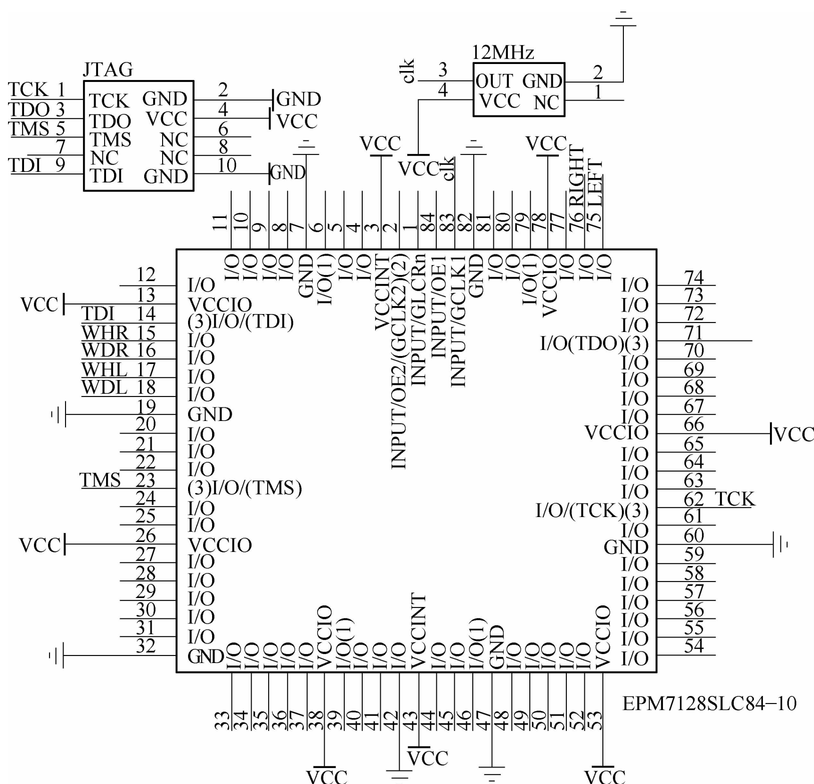



图 22-10　CPLD 控制电路

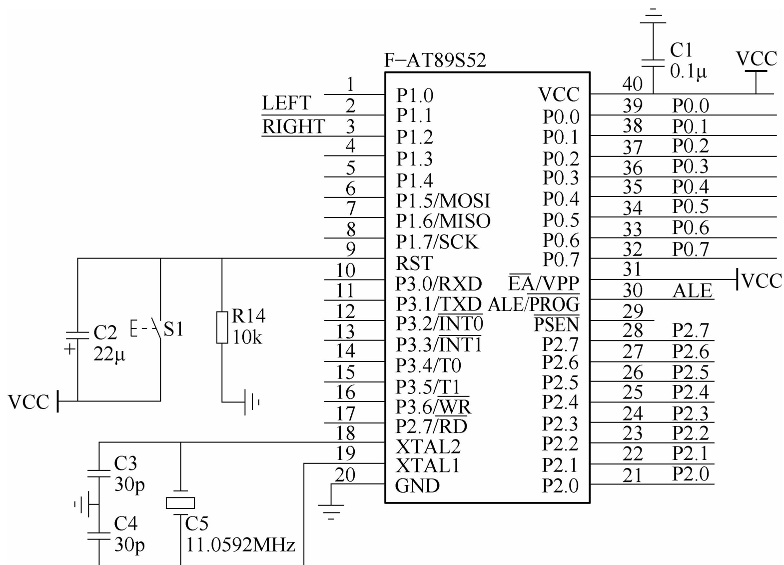



图 22-11 小车单片机电路原理图

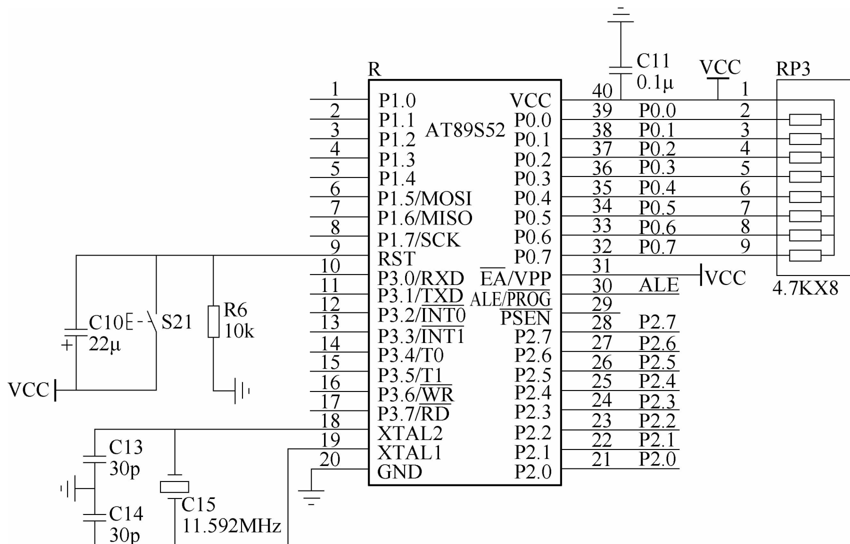



图 22-12 站台控制单片机电路原理图

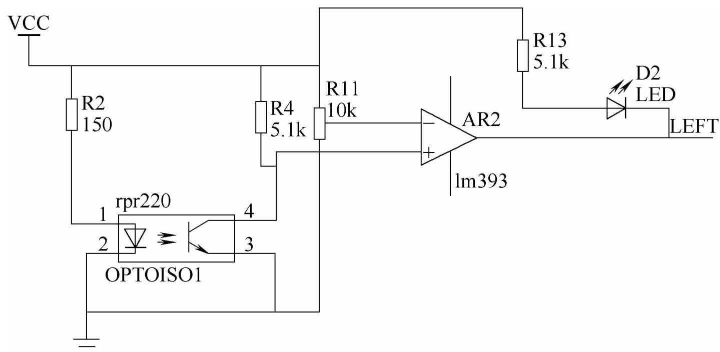



图 22-13 传感器检测与调理电路图

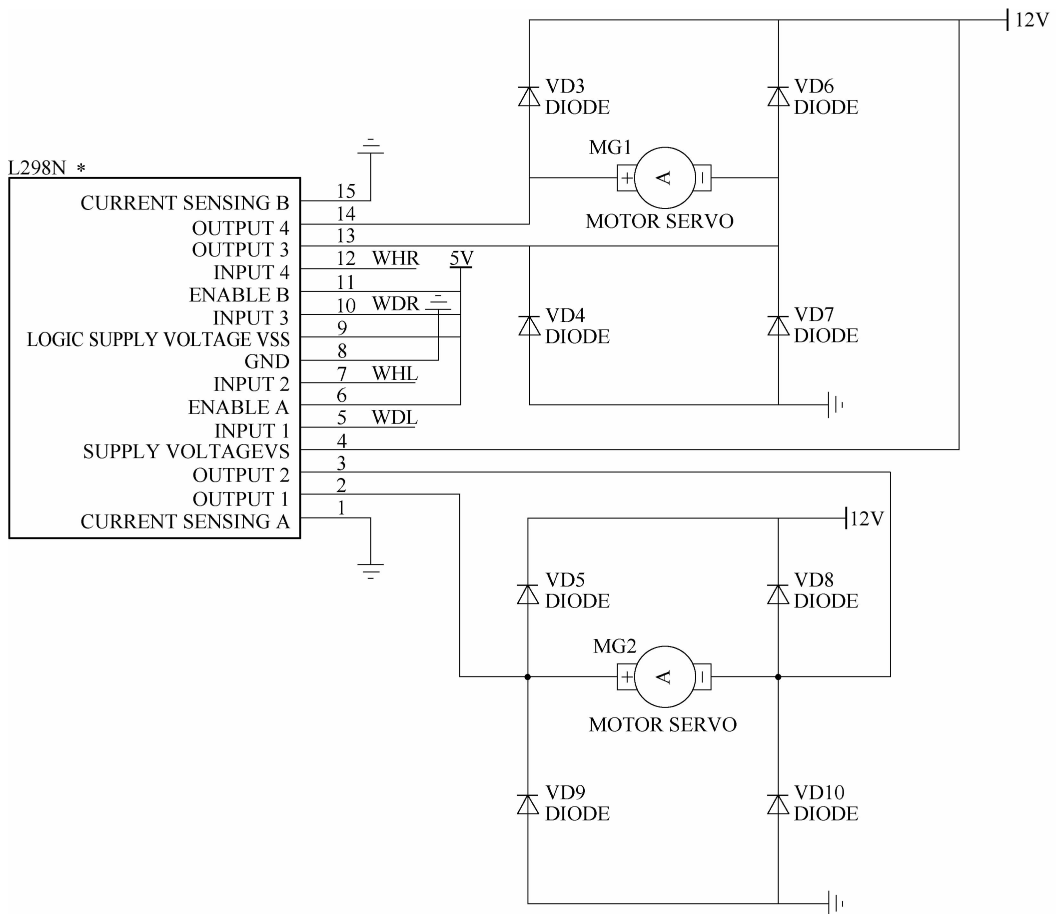



图 22-14　电动机驱动电路原理图

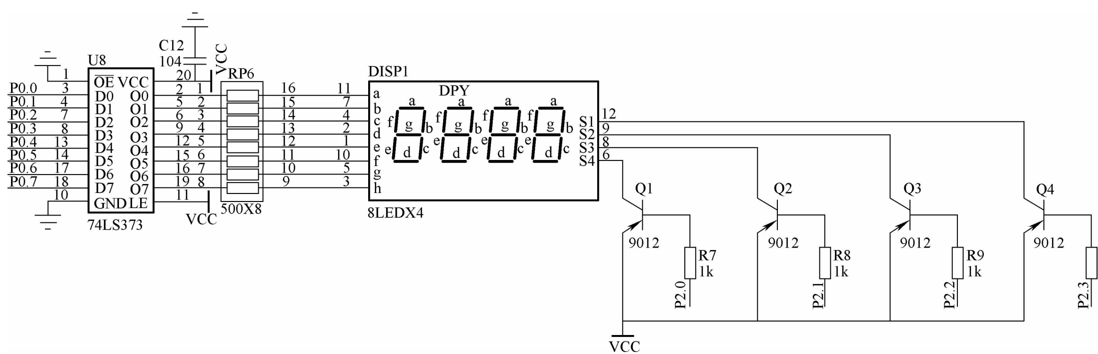



图 22-15　数码管显示电路原理图

# 第 23 章　红外遥控风扇控制系统设计

## 23.1　项目说明

### 23.1.1　研究背景

目前，国内外有很多关于智能电风扇的课题，并且很多已经取得了可观的效益。传统的电风扇大部分只能用手动换挡、调温，此外还会有一个机械定时器。这些看起来并不是很完善，功能有少。比如说会忘记及时关闭电风扇，导致的后果就是既浪费电又可能因为器件长时间工作发热而烧坏电子元器件。为了改变这些现状，有些研究者为此设计了最初的温控电风扇，由当前空间环境的温度控制风扇的开和关，在不同的温度时，控制继电器接通相应的调速挡位，以此来实现风扇的自动调速，可是它的调速方式过于机械化，不能很好地满足人们的需要。在此基础上有研究者调换了调速的方式，采用了双向的晶闸管并且通过调节电位器的阻值大小来调节转速。但是又出现了因需手动调节电位器导致使用上不方便的问题。鉴于以上几点的考虑，需要设计一种智能电风扇控制系统来解决这些问题。

本次研究的系统方案，不仅完整地将风扇的传统功能保留下来，而且还增加了外置遥控控制功能，使这款风扇操作简便，且更加安全，使风扇更具人性化、智能化。

### 23.1.2　研究方案

本课题主要任务是设计一款红外遥控风扇控制系统，该系统设计以 AT89S52 单片机为核心控制器，通过 DS18B20 温度传感器对室内环境温度进行数据采集，单片机对采集到的温度信号进行处理并输出一定占空比的 PWM，电风扇随温度变化而自动变换挡位，实现“温度高，风力大；温度低，风力弱”的性能。另外，通过红外键盘控制面板，用户可以在一定范围内设置电风扇的最低工作温度，当温度低于所设置温度时，电风扇将自动关闭；当温度高于设置温度门限值时电风扇重新启动。另外，本次设计的风扇还具有遥控的功能，并且分为“1、2、3、4”4 个不同风力大小的挡位。

红外遥控电风扇的设计分软件系统设计和硬件系统设计，主要分为 6 个模块：单片机控制模块、温度采集模块、红外发射接收模块、LCD 显示模块、温度存储模块以及电机调速控制模块，各个模块功能如下：

1）单片机控制模块：整个系统的控制核心，通过软件程序实现数据的采集、转换与输出。

2）温度采集模块：核心是温度传感器，主要用于采集室内温度，将温度的变化程度转换为相应的数字信号，实现风扇的不同控制。

3）红外发射接收模块：红外发射模块才采用现成的红外遥控器，红外接收模块主要功能是方便远距离控制风扇的运行。

4）LCD 显示模块：用于实时显示当前温度与设定的最高温度和最低值，方便用户查看。

5）温度存储模块：存储当前设置的上限温度和下限温度值，以免数据丢失。

6）电动机调速控制模块：给电动机提供相应的驱动信号，完成电动机调速，实现风扇风速调节。

根据系统功能，项目制作需要完成的工作如下：

1）用 Protel 软件设计系统的硬件电路，主要包括单片机最小系统、温度采集电路、液晶显示电路、红外发射接收电路、温度存储电路和电动机驱动电路。

2）用 Keil 软件完成单片机 AT89S52 的核心控制程序的开发，主要包括温度信号的采集与处理、液晶显示、红外按键识别与处理、温度存储程序以及 PWM 产生程序。

3）软件程序编写采用 C 语言完成。

4）对系统硬件电路和软件进行联机调试与功能完善。

## 23.2　系统概述

本系统设计主要由 6 大模块电路组成：单片机核心控制模块、LCD1602 显示模块、DS18B20 温度感应模块、红外接收以及红外发送模块、温度存储模块以及 L298N 电动机驱动模块。系统由 AT89S52 单片机核心控制，除了能实时显示当前温度外，还能通过红外遥控的方式调整温度值；温度显示采用 LCD1602 字符型液晶，它能够同时显示 16×2 即 32 个字符，该模块内部的字符发生存储器（CGROM）已经存储了 160 个不同的点阵字符图形，方便使用；温度感应模块采用的是 DS18B20，DS18B20 耐磨耐碰，体积小，使用方便，封装形式多样，适用于各种狭小空间设备数字测温和控制领域；温度存储芯片采用 AT24C02，AT24C02 是美国 Atmel 公司的低功耗 CMOS 串行 EEPROM，它是内含 256×8 位存储空间，具有工作电压宽（2.5～5.5V）、擦写次数多（大于 10000 次）、写入速度快（小于 10ms）等特点；设置红外接收模块采用集遥控信号的接收、放大、检波、整形于一身的一体化红外接头 HS0038，它能输出可以让单片机识别的 TTL 信号，大大简化了接收电路的复杂程度。其系统结构框图如图 23-1 所示。

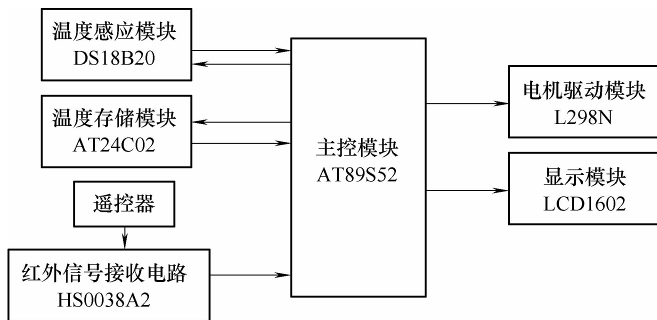



图 23-1　系统结构框图

## 23.3　系统硬件电路设计

### 23.3.1　AT89S52 单片机最小系统设计

红外线遥控风扇控制系统设计的核心是 AT89S52 单片机控制。它是一种低功耗、高性

能的 CMOS 8 位单片机，器件采用了 Atmel 公司的高密度、非易失性的存储技术制造，兼容标准 MCS-51 指令系统及 80C51 引脚的结构，适用于常规编程器。晶体振荡器选用 12MHz 的石英晶振，复位电路选用手动复位电路，方便复位单片机，其硬件电路图如图 23-2 所示。

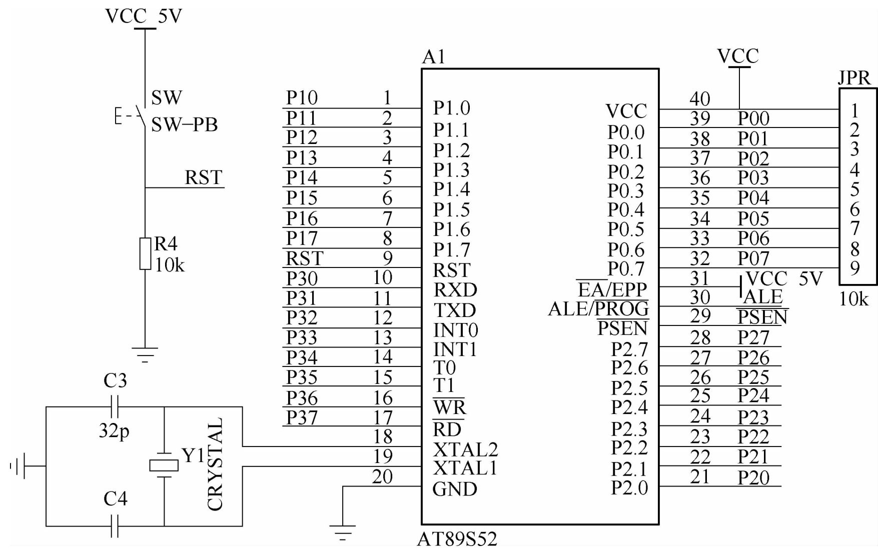



图 23-2 单片机 AT89S52 最小系统

AT89S52 有 40 个引脚和 8KB Flash 片内程序存储器，256B 的随机存储数据存储器（RAM），32 个外部双向输入/输出（I/O）端口，1 个 8 位 CPU，同时内含两个数据指针，3 个 16 位可编程定时/计数器，5 个中断优先级，一个 6 向量 2 级中断结构，两个全双工串行通信口，片内晶振及时钟电路，此外还有看门狗（WDT）电路。89S52 引脚功能如下：

VCC：电源。

GND：接地。

P0 口的每一位口线可以驱动 8 个 LSTTL 负载。在作为通用 I/O 口时，由于输出驱动电路是开漏方式，由集电极开路（OC 门）电路或漏极开路电路驱动时需外接上拉电阻；当作为地址/数据总线使用时，口线输出不是开漏的，无须外接上拉电阻。

P1、P2、P3 口的每一位能驱动 4 个 LSTTL 负载。它们的输出驱动电路设有内部上拉电阻，所以可以方便地由集电极开路（OC 门）电路或漏极开路电路所驱动，而无须外接上拉电阻。当 CPU 不对 P3 口进行字节或位寻址时，内部硬件自动将口锁存器的 Q 端置 1。这时，P3 口作为第二功能使用。

P3.0：RXD（串行口输入）。

P3.1：TXD（串行口输出）。

P3.2：INT0 外部中断 0 输入。

P3.3：INT1 外部中断 1 输入。

P3.4：T0（定时器 0 的外部输入）。

P3.5：T1（定时器 1 的外部输出）。

P3.6：WR（片外数据存储器“写”选通控制输出）。

P3.7：RD（片外数据存储器“读”选通控制输出）。

EA/VPP：访问程序存储器控制信号，当其为低电平时，对 ROM 的读操作限定在外部的程序存储器，当其为高电平时，对 ROM 的读操作是从内部存储器开始的，并可延至外部程序存储器。

ALE/PROG：编程脉冲。

PSEN：外部程序存储器读选通信号，在读外部 ROM 时，PSEN 是低电平有效，以实现对 ROM 的读操作。

RST/VPD：复位信号，当输入信号延续两个周期以上的高电平有效，用以完成单片机复位初始化操作。

XTAL1：振荡器反相放大器和内部时钟发生电路的输入端。

XTAL2：振荡器反相放大器的输出端。

XTAL：时钟晶振输入端。

在原理图 23-2 中，P0 口作为 I/O 口把数据送到 LCD 显示器上显示；P2.3 ~ P2.5 口作为 I/O 口控制 LCD 显示；P3.3 作为红外入口；P3.7 用来控制 DS18B20 温度传感器。单片机要正常工作还需要有晶振电路和复位电路，因为没有晶振电路的话，也就没有时钟周期，没有时钟周期，就无法执行程序代码，单片机也就无法工作，单片机的晶振电路是一种典型电路，如图 23-2 所示，时钟的晶振频率选择 12MHz，外接两个 32p 电容。

### 23.3.2　温度传感器电路设计

DS18B20 数字温度计是 DALLAS 公司生产的 1—Wire，即单总线器件，具有线路简单，体积小特点。因此，用它来组成一个测温系统，线路简单，十分方便。

**1. DS18B20 产品的特点**

（1）独特的单线接口仅需一个端口引脚进行通信。

（2）简单的多点分布应用。

（3）无需外部器件。

（4）可通过数据线供电。

（5）零待机功耗。

（6）测温范围 -55 ~ 125℃，以 0.5℃递增。华氏器件 -67 ~ 2570F，以 0.90F 递增。

（7）可编程的分辨率为 9 ~ 12 位，对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃。

（8）温度数字量转换时间 200ms（典型值），12 位分辨率时最多在 750ms 内把温度值转换为数字。

（9）用户可定义的非易失性温度报警设置。

（10）报警搜索命令识别并标志超过程序限定温度（温度报警条件）的器件。

（11）应用包括温度控制、工业系统、消费品、温度计或任何热感测系统。

（12）负压特性：电源极性接反时，温度计不会因发热而烧毁，但不能正常工作。

**2. DS18B20 的引脚说明**

TO-92 封装的 DS18B20 的引脚排列如图 23-3，其引脚功能描述见表 23-1。

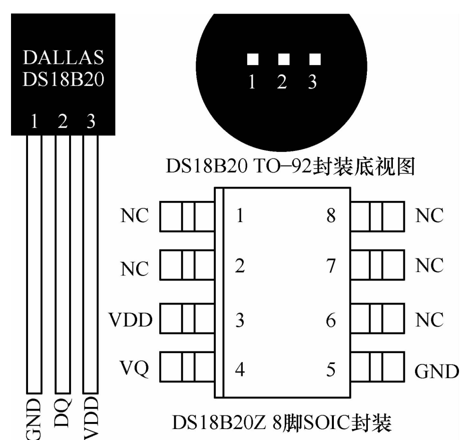



图23-3 DS18B20引脚图

表23-1 DS18B20详细引脚描述

| 序号 | 名称 | 引脚功能描述 |
|---|---|---|
| 1 | GND | 电源地 |
| 2 | DQ | 数字信号输入/输出端 |
| 3 | VDD | 外接供电电源输入端 |

**3. DS18B20使用方法**

由于DS18B20采用的是1—Wire总线协议方式，即在一根数据线实现数据的双向传输，而对AT89S52单片机来说，硬件上并不支持单总线协议。因此，我们必须采用软件的方法来模拟单总线的协议时序来完成对DS18B20芯片的访问。

由于DS18B20是在一根I/O线上读写数据，因此，对读写的数据位有着严格的时序要求。DS18B20有严格的通信协议来保证各位数据传输的正确性和完整性。该协议定义了几种信号的时序有初始化时序、读时序、写时序。所有时序都是将主机作为主设备，单总线器件作为从设备。而每一次命令和数据的传输都是从主机主动启动写时序开始，如果要求单总线器件回送数据，在进行写命令后，主机须启动读时序完成数据接收。数据和命令的传输都是低位在先。

（1）DS18B20复位时序如图23-4所示。

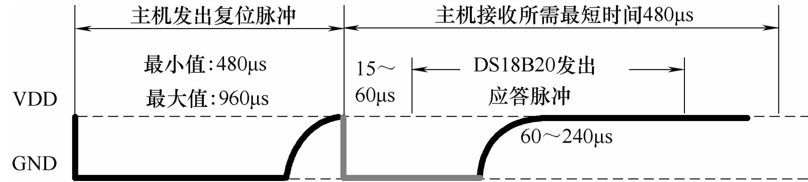



图23-4 DS18B20复位时序图

（2）DS18B20读时序 对于DS18B20的读时序分为读0时序和读1时序两个过程。DS18B20读时序是从主机把单总线拉低之后，在15s之内就得释放单总线，以让DS18B20把数据传输到单总线上。DS18B20在完成一个读时序过程，至少需要60μs才能完成。读时序如图23-5所示。

（3）DS18B20写时序 对于DS18B20的写时序仍然分为写0时序和写1时序两个过程。对于DS18B20写0时序和写1时序的要求不同，当要写0时序时，单总线要被拉低至少60μs，保证DS18B20能够在15~45μs之间能够正确地采样IO总线上的“0”电平，当要写1时序时，单总线被拉低之后，在15μs之内就得释放单总线。写时序如图23-6所示。

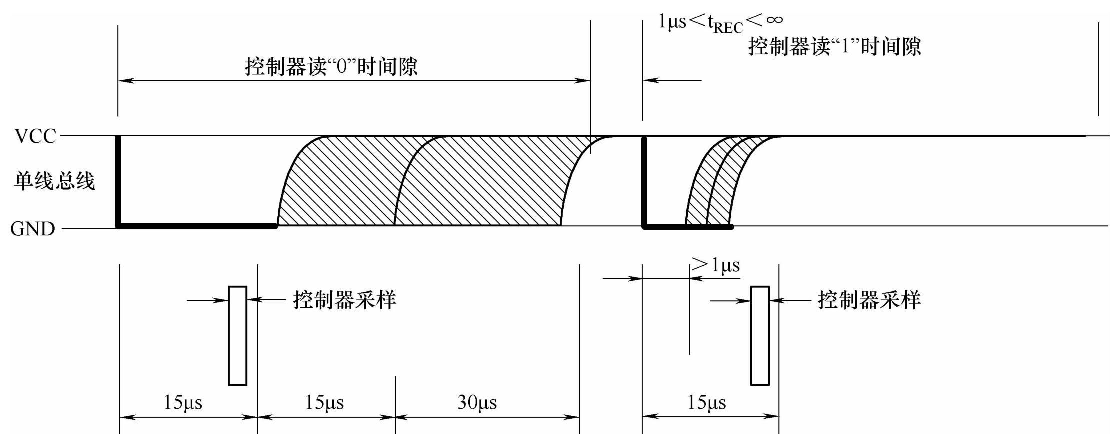



图 23-5　DS18B20 读时序

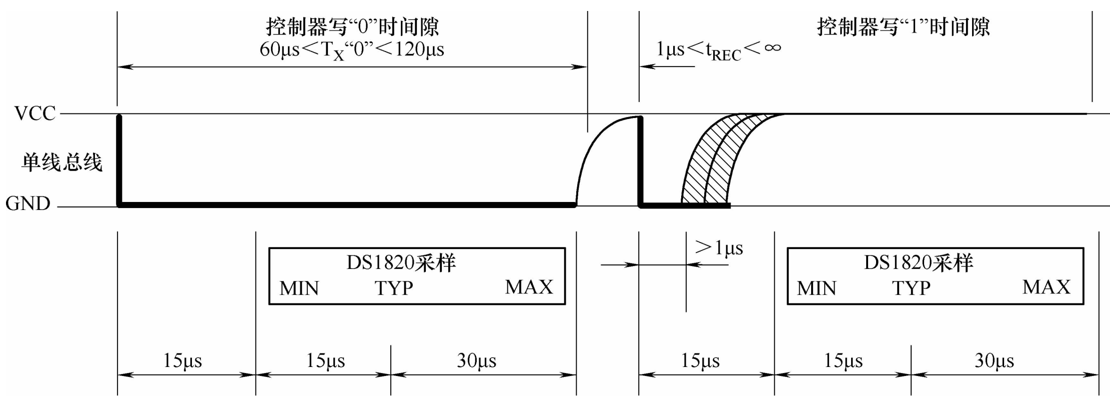



图 23-6　DS18B20 写时序

温度采集电路连接到单片机的 P3.7 引脚，I/O 口上通过 R3 上拉到 VCC。温度传感模块电路如图 23-7 所示。

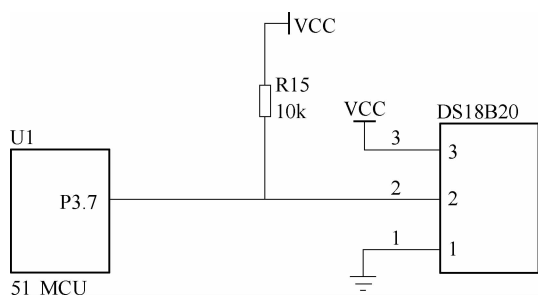



图 23-7　温度传感模块电路原理图

DS18B20 对采集的信息进行处理时，利用其内部的高速暂存存储器，其分配如图 23-8 所示。当温度转换命令发布后，经转换所得的温度值以二字节补码形式存放在高速暂存存储器的第 0 和第 1 个字节。单片机可通过单线接口读到该数据，读取时低位在前，高位在后。

表 23-2 是 DS18B20 温度采集转化后得到的 12 位数据，存储在 DS18B20 的两个 8 位的 RAM 中，二进制中的前面 5 位是符号位，如果测得的温度大于或等于 0，这 5 位为 0，

| 温度低位 | 温度高位 | TH | TL | 配置 | 保留 | 保留 | 保留 | 8位CRC |
|---|---|---|---|---|---|---|---|---|

LSB　　　　　　　　　　　　　　　　MSB

图 23-8　高速暂存分配图

只要将测到的数值乘以 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为 1，测到的数值需要取反加 1 再乘以 0.0625 即可得到实际温度。

**表 23-2 DS18B20 温度数据表**

| 温度/℃ | 二进制表示 | | | | | | | | | | | | 十六进制表示 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 符号位（5 位） | 数据位（11 位） | | | | | | | | | | | |
| 125 | 0 0 0 0 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 07D0H |
| 25.0625 | 0 0 0 0 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0191H |
| 10.125 | 0 0 0 0 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 00A2H |
| 0.5 | 0 0 0 0 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0008H |
| 0 | 0 0 0 0 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0000H |
| −0.5 | 1 1 1 1 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | FFF8H |
| −10.125 | 1 1 1 1 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | FF5EH |
| −25.625 | 1 1 1 1 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | FE6FH |
| −55 | 1 1 1 1 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | FC90H |

温度转换计算方法举例：

例如：当 DS18B20 采集到 125℃的实际温度后，输出为 07D0H，则

实际温度 = 07D0H × 0.0625 = 2000 × 0.0625 = 125℃

例如：当 DS18B20 采集到 −55℃的实际温度后，输出为 FC90H，则应先将 11 位数据位取反加 1 得 370H（符号位不变，也不作为计算），则

实际温度 = 370H × 0.0625 = 880 × 0.0625 = 55℃

## 23.3.3 LCD1602 显示模块设计

1602 液晶也叫 1602 字符型液晶，它是一种专门用来显示字母、数字、符号等的点阵型液晶模块。它由若干个 5X7 或者 5X11 等点阵字符位组成，每个点阵字符位都可以显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用。LCD 有 4 种基本操作，具体见表 23-3。

**表 23-3 LCD 与单片机之间有 4 种基本操作**

| RS | R/W | 操　作 |
|---|---|---|
| 0 | 0 | 写命令操作（初始化，光标定位等） |
| 0 | 1 | 读状态操作（读忙标志位） |
| 1 | 0 | 写数据操作（要显示内容） |
| 1 | 1 | 读数据操作（可以把显示存储区中的数据反读出来） |

LCD1602 模块的引脚如图 23-9 所示，其引脚功能如下：

RS：数据和指令选择控制端，RS = 0 命令状态；RS = 1 数据。

R/W：读写控制线，R/W = 0 写操作；R/W = 1 读操作。

A：背光控制正电源，K：背光控制地。

E：数据读写操作控制位，E 线向 LCD 模块发送一个脉冲，LCD 模块与单片机间将进行一次数据交换。

DB0 ~ DB7：数据线，可以用 8 位连接，也可以只用高 4 位连接，节约单片机资源。

VDD：电源端，VEE：亮度控制端（1 ~ 5V），VSS：接地端。

图 23-9 中单片机的 P0 口接 LCD1602 的 D0 ~ D7，作为单向数据线。单片机的 P2. 3，P2. 4，P2. 5 分别接 LCD1602 的 RS，R/W，E，作为 LCD1602 寄存器的选择、读写操作的信号线以及使能端，液晶背光接入电源 VCC。

液晶模块是用来显示存入单片机的信息，主要是通过红外遥控控制，来提取红外遥控发送的数据，然后把红外接收头获得的信息送到 LCD1602 上显示，其电路原理图如图 23-9 所示。

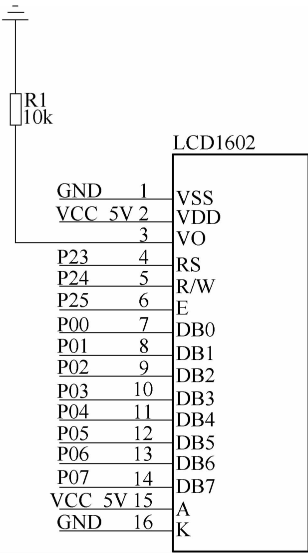



图 23-9　LCD1602 液晶显示模块电路

## 23. 3. 4　红外接收模块

红外通信是通过使用红外线光进行的通信，通常是发送设备将电信号转换成光信号，然后接收设备再将光信号转换成电信号。通用红外遥控系统由发射和接收两大部分组成，应用编/解码专用集成电路芯片进行控制操作，如图 23-10 所示。发射部分包括键盘矩阵、编码调制、LED 红外发送器；接收部分包括光、电转换放大器、解调、解码电路。

红外接收模块是用来接收红外遥控器发射的信号，并将接收到的信号输入到单片机中。在此我们采用的是 HS0038A2 一体化红外接收头。一体化红外接收头将红外接收二极管、放大、解调、整形等电路做在一起，只有 3 个引脚。一体化红外线接收器是一种集红外线接收和放大于一体，不需要任何外接元件，就能完成从红外线接收到输出与 TTL 电平信号兼容的所有工作，而体积和普通的塑封晶体管大小一样，它适合于各种红外线遥控和红外线数据传输。其电路原理图如图 23-11 所示，红外接收头的信号输出接单片机的 INT1 脚。

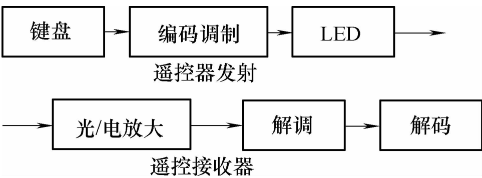



图 23-10　红外收发系统框图

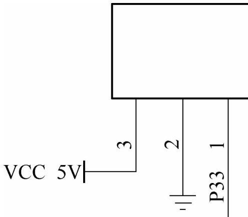



图 23-11　红外接收模块与单片机连接图

## 23. 3. 5　电动机驱动模块设计

本系统设计的电动机驱动模块主要采用 L298N 作为驱动器，通过单片机的 I/O 输入改变芯片控制端的电平，即可以对电动机进行正转、停止等操作。L298N 是 SGS 公司的产品，内部包含 4 通道逻辑驱动电路，是一种两相和四相电动机的专用驱动器，即内含两个 H 桥

的高电压大电流双全桥式驱动器，接收标准 TTL 逻辑电平信号，可驱动 46V、2A 以下的电动机。L298N 输入引脚与输出引脚的逻辑关系见表 23-4。

**表 23-4 输入引脚与输出引脚的逻辑关系**

| EA | In1 | In2 | 运转状态 |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 1 | 0 | 转动 |
| 1 | 0 | 0 | 停止 |

L298N 引脚说明如下：

5V：芯片电压 5V。

VCC：电动机电压，最大可接 50V。

GND：共地接法。

A－～D－：输出端，接电动机。

A～D＋：为步进电动机公共端，模块上接了 VCC。

ENA、ENB：高电平有效，ENA、ENB 分别为 IN1 和 IN2、IN3 和 IN4 的使能端。

IN1 ～ IN4：输入端，输入端电平和输出端电平是对应的。

电源电路采用 7805 进行电压分压，7805 也是最常用到的稳压芯片，使用非常方便，用很简单的电路即可以输入一个直流稳压电源，它的输出电压恰好为 5V，刚好是 51 系列单片机运行所需的工作电压。7805 的引脚功能如下：

1 引脚：接整流器输出的电压。

2 引脚：为公共地（也就是负极）。

3 引脚：是需要的 5V 输出电压。

图 23-12 为电源与电动机驱动电路原理图。

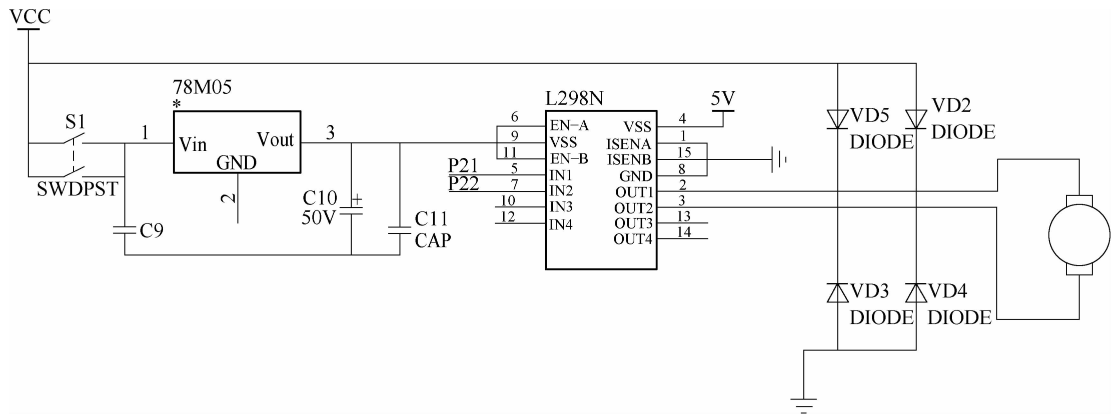



图 23-12 电源与电动机驱动原理图

## 23. 3. 6 存储电路

串行 EEPROM 因能够在线进行擦写，且断电后信息不回丢失，所以在单片机应用系统的扩展中获得广泛应用。在本设计中，采用 AT24C02 作为 EEPROM 芯片，可以使得断电重

新开启后，LCD 还可以显示之前设定的“高温”、“低温”的温度值。AT24C02 的特点是具有允许在简单的二线总线上工作的串行接口和软件协议，占用很少的资源和 I/O 线，并且支持在线编程，进行数据实时的存取十分方便。电路原理图如图 23-13 所示。

IIC 协议规定，串行时钟输入线 SCL 和串行数据/地址线 SDA 必须接上拉电阻，故电路通过两个 10k 电阻上拉到电源，其中 SCL 连到 P3.4，SDA 连到 P3.5。A0，A1，A2 为器件地址输入端，因为在这个 $I^2C$ 总线上只有一个器件，所以把 AT24C02 的地址设为 000，即把 A0、A1、A2 都接地。WP 为写保护端接地，允许芯片进行读写操作。

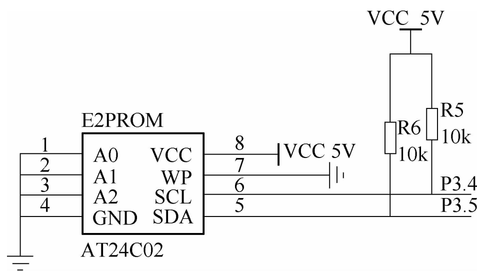



图 23-13　温度存储电路原理图

## 23.4　系统软件设计

本系统软件程序设计主要由主程序、温度采集子程序、红外解码程序、LCD 同步显示子程序、温度存储子程序、以及电动机驱动子程序等模块组成。其中主程序主要起到一个导向和决策的功能，进入系统时，LCD 首先显示当前的温度；温度感测子程序核心是温度的采集，主要用于采集室内温度，用温度的变化程度发出相应的控制信号；红外解码子程序主要实现红外遥控器上发送的信号解码；LCD 子程序主要是显示当前温度感测、设置最低温度、设置的最高温度等温度信息；温度存储子程序将设置的温度值存储到 IIC 芯片中；电动机驱动子程序主要实现当前电动机的“温度高，风力大；温度低，风力弱”的性能。下面对这几个模块的工作原理和流程设计详细说明如下。

### 23.4.1　主程序设计

主程序主要是读取 DS18B20 温度，并通过 LCD1602 显示出来。首先对红外编码和 LCD 进行初始化，调用 init_sys（）函数；由于红外遥控的控制是通过中断的方式，因此我们对单片机的中断入口进行设置；最后调用 Init_DS18B20（）函数来读取当前温度，在 LCD 液晶显示屏上显示字符串。其流程如图 23-14 所示。

关键程序代码如下：

```
init_sys();                  //初始化
CodeTemp = 0;                //初始化红外编码字节缓存变量
ZKB1 = 50;                   //占空比初值
motorsw = 0;
ET0 = 0;                     //关定时器 0 中断
i = ReadTemperature();  //读取当前温度
IRcvStr(at24c02,0x01,temp_rw_buff,0x05);      //读取 IIC 里储存的温度数据
if(temp_rw_buff[4] == 0xdf){
    temp_high = temp_rw_buff[0] * 0x100 + temp_rw_buff[1];   //还原数据
    temp_low = temp_rw_buff[2] * 0x100 + temp_rw_buff[3];   //还原数据
```

```
    }
else{
    temp_high = 300;
    temp_low = 100;
    }
```

## 23.4.2 温度采集子程序设计

温度采集子程序的任务是读写温度信息，图 23-15 是关于 DS18B20 子程序的流程图。具体关于 DS18B20 的时序内容可以参考 23.3.2 节内容，DS18B20 初始化子程序，向 DS18B20 读一字节数据，向 DS18B20 写一字节数据，向 DS18B20 读温度值。

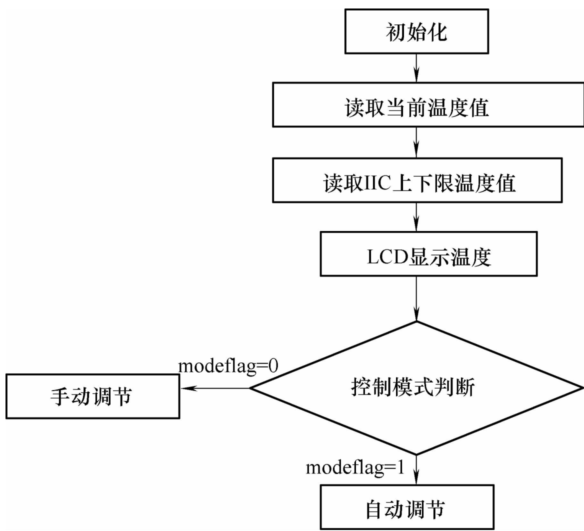



图 23-14　主程序流程图

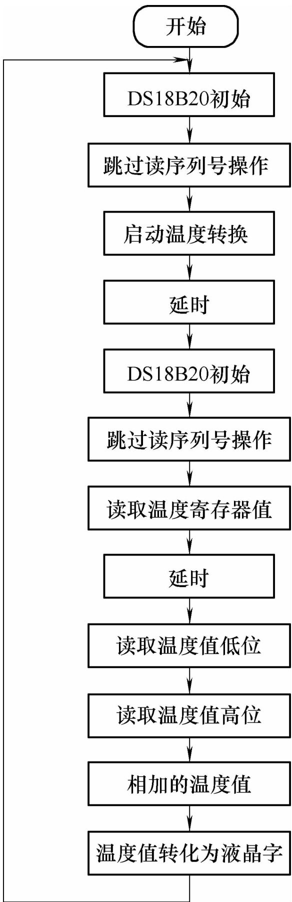



图 23-15　温度采集子程序流程图

关于向 DS18B20 读温度值的关键程序代码说明如下：

```
ReadTemperature(void)
{   Init_DS18B20();              //初始化
    WriteOneChar(0xcc);          //跳过读序列号的操作
    WriteOneChar(0x44);          //启动温度转换
    delay1(125);                 //转换需要一点时间,延时
    Init_DS18B20();
```

```
WriteOneChar(0xcc);
WriteOneChar(0xbe);      //读温度寄存器(头两个值分别为温度的低位和高位)
tempL = ReadOneChar(); //读出温度的低位 LSB
tempH = ReadOneChar();//读出温度的高位 MSB
//温度转换，把高低位做相应的运算转化为实际温度
temperature = ((tempH * 256) + tempL) * 0.0625;
delay1(200);
return(temperature * 10 + 0.5);}
```

### 23.4.3　红外接收程序设计

当我们按下遥控器的按键时，遥控器将发出一串二进制代码，我们称它为一帧数据。根据各部分功能。可将它们分为 5 部分，分别为引导码、地址码、地址码、数据码、数据反码。遥控器发射代码时均是低位在前，高位在后。在本次设计中，红外接收子程序的主要功能就是接收来自红外遥控器发出的信号，以实现 LCD 显示。

其外部中断 INT0 的关键代码如下：

```
void int1(void) interrupt 2 using 2
{uchar k,i,j;
    EA = 0;                  //跳入中断,但仍可对 P3.2(INT0)进行电平变化的读取
    for(k = 0;k < 10;k ++)
    {Delay0_9ms();
    if(IRsignal == 1)        //如果 0.9ms 后 IRsignal = 1,说明不是引导码
    {   k = 10;
    break;
    else if(k == 9)          //如果持续 10 ×0.9ms = 9ms 的低电平,说明是引导码
    {while(IRsignal == 0);
    Delay4_5ms();            //跳过持续 4.5ms 的高电平
    for(i = 0;i < 4;i ++)    //分别读取 4 个字节
    {for(j = 1;j < = 8;j ++) //每个字节 8 位的判断
    {while(IRsignal == 0);   //等待上升沿此处用得很好:因为 0.56ms 的低电平(接收时)是代码 0 与 1 的相同部分
    Delay0_9ms();            //从上升沿那一时刻开始延时 0.9ms(因为 0.9 介于 0.565( =1.125 - 0.56)与 1.69( =2.25 - 0.56)之间),再判断 IRsignal
    if(IRsignal == 1)        //如果 IRsignal 是"1",高位置"1",并向右移一位
    {Delay1ms();             //为什么要延时 1ms 呢? 因为要使 IRsignal 跳至低电平(即 0.56ms 的 0 与 1 相同部分上)
    CodeTemp = CodeTemp|0x80; //此处的算法很好
    if(j < 8) CodeTemp = CodeTemp >> 1;}
    else
    if(j < 8)
```

```
    CodeTemp = CodeTemp >> 1;  //如果 IRsignal 是"0",则直接向右移一位,自动补"0"
    }
    IRcode[i] = CodeTemp;
    CodeTemp = 0; }
    DelayIR(); }
    EA = 1; }
    EA = 1; }
```

### 23.4.4　LCD 显示子程序设计

在 LCD 显示程序之前，必须对 LCD 进行初始化，否则液晶显示屏无法进行正常工作。在对液晶模块的初始化中要先设置其显示模式，在液晶模块显示字符时光标是自动右移的，无需人工干预。LCD 初始化程序如下：

```
lcd_init()
{   lcd_wcmd(0x38);
    delay(1);
    lcd_wcmd(0x0c);
    delay(1);
    lcd_wcmd(0x06);
    delay(1);
    lcd_wcmd(0x01);
    delay(1);
}
```

从通电开始延时，先经过判忙后再进行功能设置，过一段时间后可以设置显示状态（如设置行、位或阵列），才可以设置输入方式，实现一个字符的写入过程关键程序代码如下所示：

```
void lcd_char_write(uchar x_pos,y_pos,lcd_dat)//LCD1602 字符写入
{   x_pos &= 0x0f;                //X 位置范围 0 ~ 15
    y_pos &= 0x01;                //Y 位置范围 0 ~ 1
    if(y_pos == 1) x_pos += 0x40;
    x_pos += 0x80;
    lcd_wcmd(x_pos);
    delay(5);
    lcd_bz();
    rs = 1;
    rw = 0;
    ep = 0;
    P0 = lcd_dat;
    delay(5);
    ep = 1;
```

```
ep = 0; }
```

## 23.4.5　电动机驱动子程序设计

脉宽调制（PWM）的控制方式就是对逆变电路开关器件的通断进行控制，使输出端得到一系列幅值相等的脉冲，用这些脉冲来代替正弦波或所需要的波形。也就是在输出波形的半个周期中产生多个脉冲，使各脉冲的等值电压为正弦波形，所获得的输出平滑且低次谐波少。按一定的规则对各脉冲的宽度进行调制，即可改变逆变电路输出电压的大小，也可改变输出频率。采用 PWM 进行调速。PWM 信号的产生程序所示如下：

```
void timer0(void)interrupt 1 using 2     //定时器 0 中断
{static uchar click = 0;                 //中断次数计数器变量
  TL0 = 0x91;                            //恢复定时器初始值 10μs 的初始值是 11.0596MHz
  TH0 = 0xFF;
  ++click;
  if(click >= 100)click = 0;
  if(click <= ZKB1)   //当小于占空比值时输出低电平，高于时是高电平，从而实现占
                      //空比的调整
    motor = 0;
  else
    motor = 1;}
```

当实际温度低于设定的温度区域，那么风扇将关闭；当实际温度处于设定的温度区域内，那么风扇开启中档风速；当实际温度高于设定的温度区域，那么风扇开启高档风速。具体实现过程如下列程序所示：

```
while(1)
{IRcodeHandle() ;
i = ReadTemperature();       //读取当前温度
if(modeflag)
    {if(temp_high < i&&IRcode[2]! = 0x00)
        {motorsw = 1;
        ET0 = 1;
        ZKB1 = 100;       //开电动机  高风
    }
    if(temp_low > i&&IRcode[2]! = 0x00)
        {motorsw = 0;
        ET0 = 0; //关电动机}
    if(temp_high > i&&temp_low < i&&IRcode[2]! = 0x00)
    {motorsw = 1;
    ET0 = 1;
    ZKB1 = 50;              //开电动机  中风
    }}
```

四档风速的关键程序如下：

```
void IRcodeHandle( )
{switch(IRcode[2])
{case 0x00:
        motor =1;               //pwm 关闭
        motorsw =0;
        ET0 =0;                 //关定时器 0 中断 关闭电动机
        break;
    case 0x11:
        motorsw =1;
        ZKB1 =25;               //调占空比
        ET0 =1;                 //开定时器 0 中断
        break;
    case 0x12:
        motorsw =1;
        ZKB1 =50;
        ET0 =1;                 //开定时器 0 中断
        break;
    case 0x13:
        motorsw =1;
        ZKB1 =75;
        ET0 =1;                 //开定时器 0 中断
        break;
    case 0x14:
        motorsw =1;
        ZKB1 =100;
        ET0 =1;                 //开定时器 0 中断
        break;
    case 0x28:
        modeflag =0;
        break;                  //手动控制
    case 0x29:
        modeflag =1;
        break;                  //自动控制
}
}
```

### 23.4.6 温度存储子程序设计

温度存储子程序设计包括起动总线函数、结束总线函数、发送字节数据传送函数、接收

字节数据传送函数、应答子函数、向有子地址器件发送多字节数据函数和向有子地址器件读取多字节数据函数等部分。在此，主要对向有子地址器件发送多字节数据函数和向有子地址器件读取多字节数据函数的部分进行说明。

向有子地址器件发送多字节数据函数是启动总线、发送地址、发送子地址及数据、结束总线的全过程。器件地址 sla、子地址 suba 发送的内容为 s 指向的内容，发送 no 个字节。如果返回 1 表示操作成功，否则操作有误。具体实现过程如下列程序所示：

```
bit ISendStr(uchar sla,uchar suba,uchar *s,uchar no)
{
    uchar i;
    if(no==0) return(1);
    Start_I2c();                    //启动总线
    SendByte(sla);                  //发送器件地址
    if(ack==0)return(0);
    SendByte(suba);                 //发送器件子地址
    if(ack==0)return(0);
    for(i=0;i<no;i++)
    {
        SendByte(*s);               //发送数据
        if(ack==0)return(0);
        s++;}
        Stop_I2c();                 //结束总线
        return(1);
    }
}
```

向有子地址器件读取多字节数据函数是启动总线、发送器件地址、发送子地址、读取数据，结束总线的全过程。将器件地址 sla、子地址 suba 的读出的内容放入 s 指向的存储区，读 no 个字节。如果返回 1 表示操作成功，否则操作有误。具体实现过程如下列程序所示：

```
bit IRcvStr(uchar sla,uchar suba,uchar *s,uchar no)
{
    uchar i;
    Start_I2c();                    //启动总线
    SendByte(sla);                  //发送器件地址
    if(ack==0)return(0);
    SendByte(suba);                 //发送器件子地址
    if(ack==0)return(0);
    Start_I2c();
    SendByte(sla+1);
    if(ack==0)return(0);
    for(i=0;i<no-1;i++)
```

```
{    * s = RcvByte( );              //发送数据
     Ack_I2c(0);                   //发送就答位
     s ++ ;}
  * s = RcvByte( );
  Ack_I2c(1);                      //发送非应位
  Stop_I2c( );                     //结束总线
  return(1);
}
```

## 23.5 系统调试与总结

### 23.5.1 系统调试

（1）采用 Protel 软件设计电路的原理图，并制作 PCB 板，完成硬件电路的焊接与测试。

（2）焊接其他器件，如 LCD、温度传感器、红外接收器、电动机、扇叶等。安装完成后，如图 23-16 所示。

（3）系统功能测试　通过 USB 接口连接电源。电路上电会显示出 Temp 实际温度及系统初始化时的最高、最低温度。如图 23-17 所示。

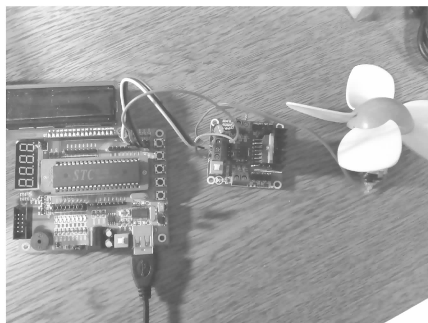



图 23-16　红外线遥控风扇控制系统整体实物图

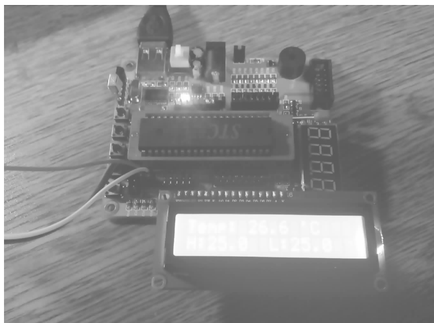


图 23-17　系统上电界面

（4）遥控器按键设置　红外遥控器作为本次设计的红外线发射端。其中“开始键”可以“启动”风扇和“关闭”风扇。F1 为“手动”，可以手动调节风扇转速大小；F2 为“自动”，即可以设置“温度高，风力大；温度低，风力弱”的性能；“1”、“2”、“3”、“4”按键为在手动情况下选择风扇风力的大小。“左键”是选中“高温数值”；“右键”是选中“低温数值”。“上键”、“下键”为调节温度值的“大”、“小”；“确认”为“确定数值”，如图 23-18 就是本设计使用的红外遥控器。

通过红外遥控器，选中“F1”手动调节电风扇，并且调节“H”和“L”的数值，使实际温度处于两只之间，如图 23-19 所示。

因为此时实际温度处于高温和低温之间，所以电机启动，扇叶转动，如图 23-20 所示。



图 23-18　红外遥控器

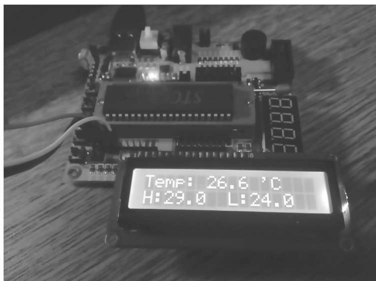



图 23-19　通过红外遥控改变数值



图 23-20　扇叶转动

### 23.5.2　系统总结

（1）LCD 显示问题　每次刷屏时，会有延迟现象产生，改进方法就是要调节好延时功能。

（2）红外遥控问题　红外遥控在设置按键的时候必须设置充分的延迟以消除按键抖动，否则无法正常按键。

（3）DS18B20 问题　DS18B20 在第一次使用的时候必须初始化后才能正常使用。

（4）电源问题　在理论上，电动机的电源应该是独立使用 7805 稳压芯片供电的，但是由于在实际的操作中存在设计的不足，所以最终全部采用 UBS 端进行供电。

（5）电动机问题　在对高档的时候，电动机会因为电压不够导致不能转动，解决方案就是提供足够的电源。

## 附件：系统设计的电路原理图

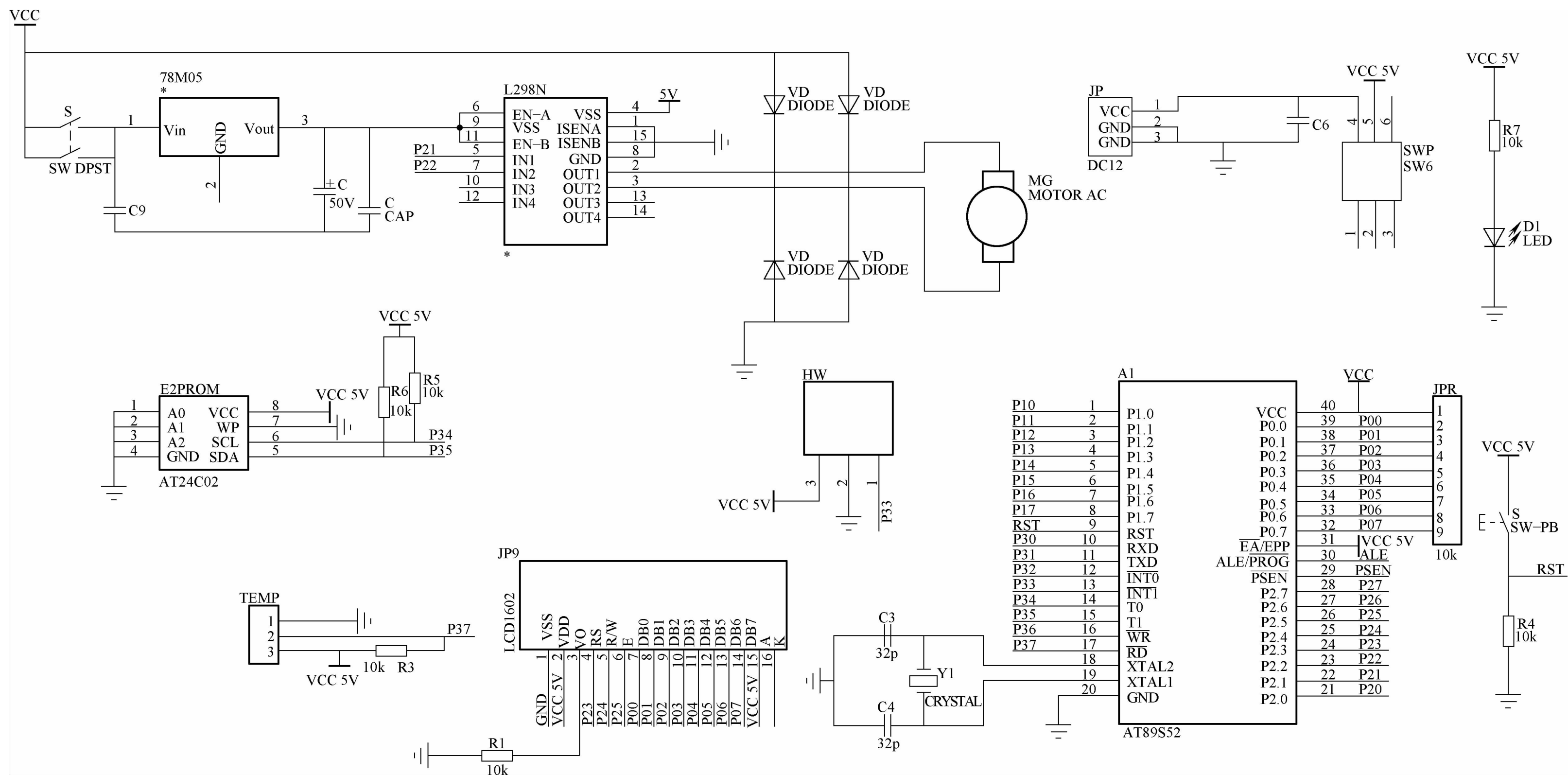



图23-21　系统设计电路原理图

# 第 24 章　多功能微电脑模拟电子秤设计

## 24.1　项目说明

### 24.1.1　项目背景

电子衡器是配有电子称重装置的仪器，主要由称重传感器、承载器、称重仪表三部分组成。其主要工作原理是将作用在称重承载器上的力传递到称重传感器上，并将其转换成与之对应的模拟信号（电压或电流），输出的模拟信号通过 A-D 转换器转换成数字信号，并在仪表盘上显示出来。

我国从 20 世纪 60 年代中期开始研制和生产电子秤，早期的电子秤是模拟指针式，20 世纪 80 年代中后期发展为数字式，20 世纪 90 年代末至 21 世纪初已研制开发出微机式产品。而数字化传感器的推出则是在 20 世纪 80 年代末，美国 SENSORTRONICS（STS）公司在 1988 年全美衡器展览会上首次推出了一种将 A-D 转换器集成在普通应变式称重传感器中，即数字传感器，这种传感器在美国市场备受瞩目。

应变式称重传感器生产工艺环节比较多，工艺比较复杂，尽管投入大量人力、物力及精力，进行温度、零点、灵敏度、非线性、滞后、蠕变等性能的补偿及四角误差调整，而补偿技术设备却仍难以达到高质量要求，普遍存在着生产成本和质量成本一直居高不下的严重问题。但是随着电子技术的飞跃进步，单片机的普及推广，数字化称重传感器应运而生。单片机，亦称单片微电脑或单片微型计算机，它是把中央处理器（CPU）、随机存储器（RAM）、只读存储器（ROM）、输入/输出端口（I/O）等主要计算机功能部件集成在一块集成电路芯片上。将传统应变式称重传感器与现代电子技术相结合实现新型电子称量技术，将模拟重量信号进行 A-D 转换预处理，自动对应变式称重传感器的零点、温度、非线性、滞后、蠕变性能等进行数字化补偿，最后输出数字信号。

目前，单片机已成为智能仪器仪表、工业测控、计算机网络与通信设备领域中最为理想的控制处理器。采用单片机技术进行物体重量信号的采集与计算，已经成为目前比较流行的一种新型电子称发展方向。

### 24.1.2　设计总体方案论证

由于系统设计的是一个电子秤称重电路，生活中物体的重量信号是通过称盘下的重量电阻应变式传感器产生电信号，但是为了方便电路调试，在调试过程中用电位器来产生电信号，根据电子秤的电位器有 3 个接点，在首尾端加上一定的电位差后，滑动接点在不同的位置上就有不同的电位，从而产生了不同的电信号。然后送入 A-D 转换芯片进行 A-D 转换，将模拟电压信号转换成数字量，转换后的数字量与电信号（物重）成正比，再进入 89S51 单片机经过数据处理，单片机产生一组满足显示要求的数据，送至显示电路显示出实际重

量。另一方面，商品单价通过键盘电路送入 89S51 单片机，物重与单价经过数据处理运算产生总价，在显示电路上显示出来，电源电路提供各模块电路的工作电压。

## 24.2 多功能微电脑电子秤实现原理

根据任务要求，选用 89S51 单片机为核心，组成多功能微电脑电子秤的称量电路。系统主要有 89S51 单片机、A-D 转换电路、按键电路、显示电路、复位电路、电源电路构成。通过按下不同的按键，单片机接收到不同的指令，实现不同的功能。其系统设计的结构框图如图 24-1 所示。

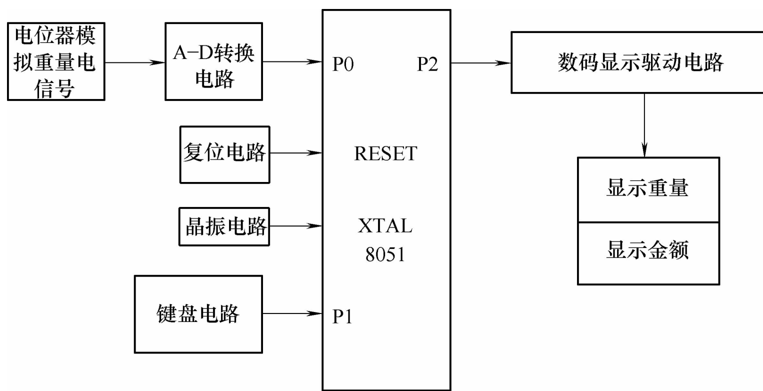



图 24-1 模拟电子秤结构框图

由电源给电位器供电，并将此电信号送给 A-D 转换电路处理后送给单片机，再结合键盘电路的电信号（采用 4 个中断式键盘，按键 1 是确定键，从 ADC0809 采集的数据重量与单价相乘，计算出金额，并把重量、金额送到数码管显示输出；按键 2 是自加键；按键 3 是自减键；按键 4 有两个功能，移位和清零）实现价格与总价的控制。4 个按键与 74LS47 相连接，使用其中的 3 个与门与 51 单片机的 INT0 连接。无论哪一个按键按下进入按键中断服务子程序，完成按键操作实现的功能。

## 24.3 微电脑电子秤硬件电路设计

### 24.3.1 51 单片机最小系统

单片微型计算机简称单片机。由于它的结构及功能均按工业控制要求设计，所以又称单片微控制器（Single Chip Microcontroller）。只要外加少许电子零件便可以构成一套简易的计算机控制系统，故又称单片微型计算机（Single Chip Microcomputer）。从 1976 年 8 位单片微机诞生，至今已发展有 16 位，32 位单片微机，但一直是以 8 位机为主流机型。

MCS-51 系列单片机是 Intel 公司继 MCS-48 之后最早推出的增强型 8 位单片机，它是国内外单片机应用的一个主流产品。该系列产品在我国应用最为广泛。MCS-51 系列单片机作为单片机品种的典型代表。

单片机的各部分功能如下：

（1）中央处理器（CPU）　中央处理器是单片机的核心，完成运算和控制功能。MCS-51 的 CPU 能处理 8 位二进制数或代码。

（2）内部数据存储器（内部 RAM）　89S51 芯片中有 128 个 RAM 单元，用于存放可读写的数据，简称内部 RAM。

（3）内部程序存储器（内部 ROM）　89S51 芯片内部有 8KB 掩膜 ROM，用于存放程序或表格，因此称为程序存储器。简称内部 ROM。

（4）定时/计数器　89S51 共有两个可编程的 16 位定时/计数器，以实现定时或计数为功能。

（5）并行 I/O 口　MCS-51 共有 4 个可编程的 8 位的 I/O 口（P0，P1，P2，P3），以实现数据的并行输入/输出。

（6）串行口　MCS-51 单片机有一个全双工的串行 I/O 口，以实现单片机和其他设备之间的串行数据传送。

（7）中断控制系统　89S51 单片机具有 5 个中断源，两个中断优先级的中断控制系统，以满足控制应用的需要。

51 单片机内部有振荡和时钟电路，但石英晶体和微调电容需外接。时钟电路为单片机产生时钟脉冲序列。系统允许的晶振频率为 2～12MHz。

89S51 是标准的 40 引脚双列直插式集成电路芯片，引脚排列如图 24-2 所示。

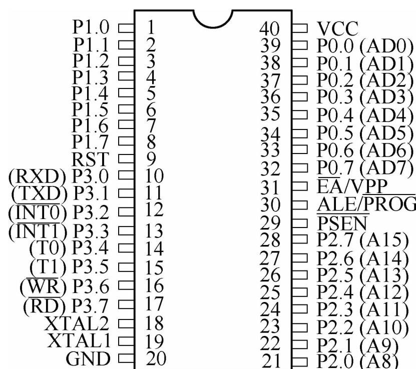



图 24-2　89S51 引脚图

（1）P0.0～P0.7　P0 口是一个 8 位双向 I/O 口。

（2）P1.0～P1.7　P1 口是一个带有上拉电阻的 8 位双向 I/O 口。

（3）P2.0～P2.7　P2 口是一个带有上拉电阻的 8 位双向 I/O 口。访问外部存储器时，送出地址高 8 位。

（4）P3.0～P3.7　P3 口是一个带有上拉电阻的 8 位双向 I/O 口。双功能接口，①能作为通用的 I/O 口，②作为特殊信号线使用。

（5）ALE　地址锁存控制信号。

（6）PSEN　外部程序存储器读选通信号。

（7）EA　访问内外程序存储齐控制信号。

（8）RST　复位信号。

（9）XTAL1 和 XTAL2　外接晶体引线端。

（10）VSS　地线。

（11）VCC　5V 电源。

单片机若能工作运行，其工作的最小系统电路如图 24-3 所示，单片机最小系统由主控芯片 51 单片机、晶振电路和复位电路构成。其中，晶振电路通过单片机的 18、19 引脚的 XTAL1 和 XTAL2 外接电容和石英晶体产生时钟信号，供单片机工作运行，复位电路采用手动复位。

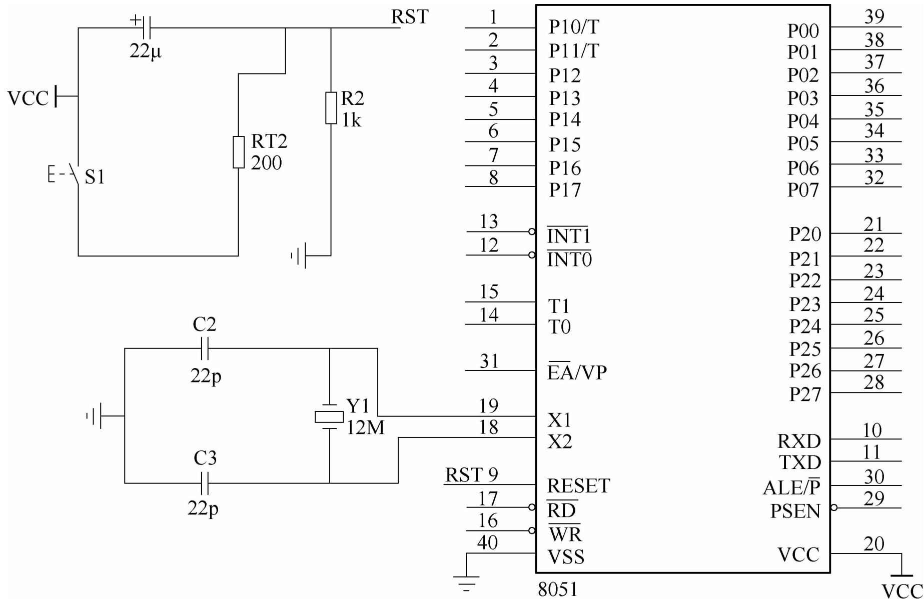



图 24-3　51 单片机最小系统电路

## 24.3.2　键盘电路

图 24-4 中，采用的是中断式键盘。一共有 4 个按键，键 1 是确定键，从 ADC0809 采集的数据重量与单价相乘，计算出金额，并把重量、金额送到显示模块显示；键 2 是自加键；键 3 是自减按键；键 4 有两个功能，移位和清零。4 个按键与 74LS08 相连接，使用其中的 3 个与门与 8051 的 INT0 连接。无论哪一个按键按下就进入中断。

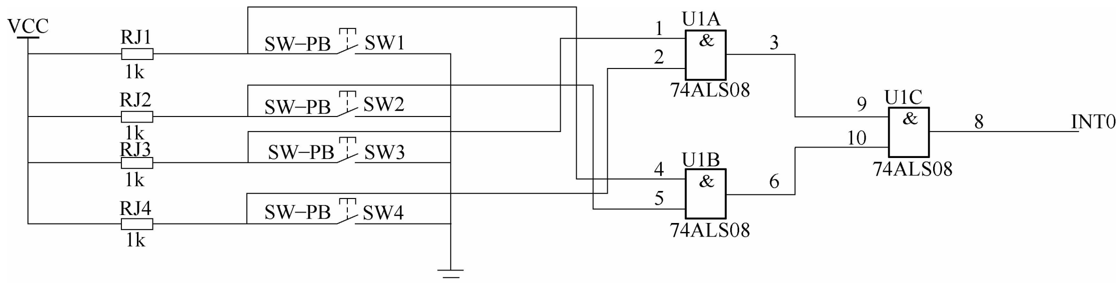



图 24-4　键盘电路图

## 24.3.3　ADC0809 接口电路

图 24-5 中，51 单片机的 P2.7 引脚、RD、WR 与 74LS02 的两个或非门相连接，输出的信号接到 ADC0809 的地址锁存允许 ALE、输出允许 ENABLE、转换启动信号 START 中，启动 ADC0809 开始工作；8 路模拟开关的地址选择线 ADDA、ADDB、ADDC 均接地，选中通道 IN－0，IN－0 与电位器相连，电路调试过程中通过滑动电位器产生不同电压信号来模拟称重的重量信号，D0～D7 与 51 单片机的 P0 口相连，对 A-D 转换后的信号进行处理。

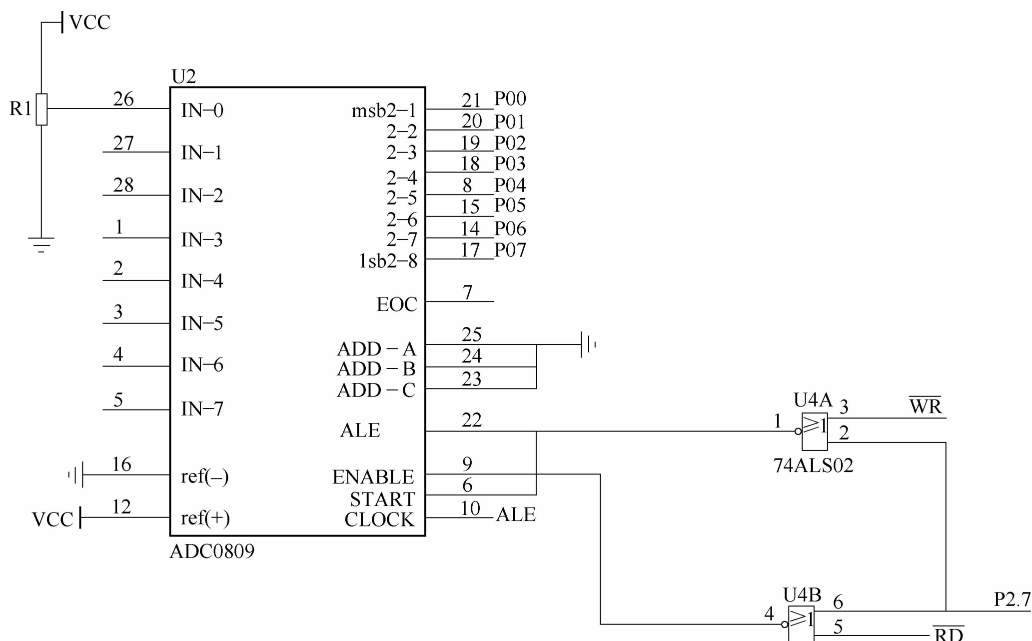



图 24-5　ADC0809 接口电路图

## 24.3.4　数码显示电路

图 24-6 中 51 单片机的 P0.0，P0.1，P0.2，P0.3 与 74LS47（即 BCD—7 段译码驱动器）的 A、B、C、D 4 个 BCD 码输入端相连，74LS47 的试灯输入 $\overline{LT}$、灭灯输入/脉动灭灯输出 BI/RBO 和 RBI 脉动灭灯输入接高电平时，使得所有段的输出端都打开，74LS47 输出端 a、b、c、d、e、f、g 则与 8 个数码管段码端相连，实现单价、总价和重量信号的实时显示。

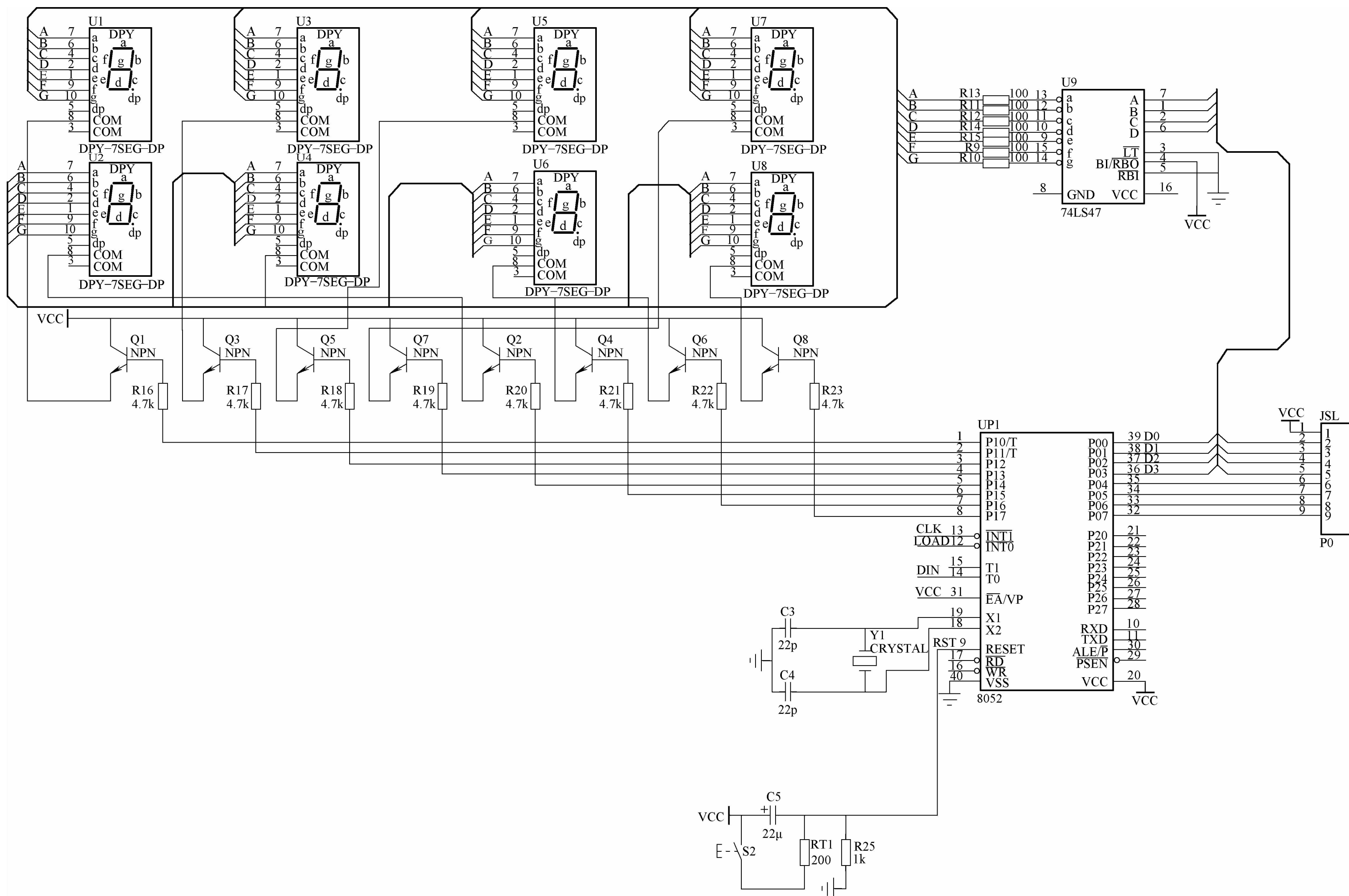

U1
U2
U3
U4
U5
U6
U7
U8
DPY
DPY-7SEG-DP
COM
VCC
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
NPN
R16
R17
R18
R19
R20
R21
R22
R23
4.7k
U9
R13
R11
R12
R14
R15
R9
R10
100
LT
BI/RBO
RBI
GND
74LS47
UP1
P10/T
P11/T
P12
P13
P14
P15
P16
P17
P00
P01
P02
P03
P04
P05
P06
P07
39 D0
38 D1
37 D2
36 D3
CLK
LOAD
INT1
INT0
DIN
T1
T0
EA/VP
X1
X2
RST
RESET
RD
WR
VSS
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
8052
JSL
P0
C3
C4
22p
Y1
CRYSTAL
C5
22μ
S2
RT1
200
R25
1k

图24-6　数码显示电路

## 24.4　微电脑电子秤软件实现

单片机的软件设计部分采用 C51 编程，在 Keil C51 开发环境下对设计的代码进行编译、仿真和调试。

### 24.4.1　主程序设计

整个系统采用 C 语言编程。主要采用中断程序设计来完成系统的要求，软件程序的主要任务有重量、金额的显示、清零及过载显示。

软件程序包括主程序、显示程序、中断程序，A-D 中断服务程序完成采样数据的存储，键盘中断服务程序完成键盘的扫描，用于输入物品的金额，重量。其程序流程如图 24-7 所示。

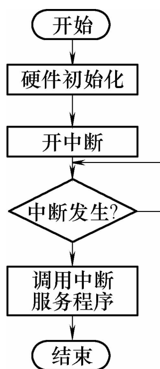



图 24-7　主程序流程图

主程序关键程序如下：

```
void main(void)
{
    EA = 1; IT0 = 1; EX0 = 1;
    LOAD = 1; CLK = 1;
    P2 = 0; delay(30000);
    transfer(0,10);
    while(1);
}
void transfer(uchar prm1,uchar prm2)
{
    uchar i = 0; LOAD = 1;
    for(i = 0;i < 8;i ++ )
    {
        if((prm1&0x01) == 1){ DIN = 1; CLK = 1; delay(1); CLK = 0; }
        else { DIN = 0; CLK = 1; delay(1); CLK = 0; }
```

```
        prm1 = prm1 > > 1;
        delay(10);//延时时间不能太短,否则接收数据错误
    }
    for(i = 0;i < 8;i ++ )
    {
        if( (prm2&0x01) == = 1) { DIN = 1; CLK = 1; delay(1); CLK = 0; }
        else { DIN = 0; CLK = 1; delay(1); CLK = 0; }
        prm2 = prm2 > > 1;
        delay(10);
    }
    LOAD = 0;
    delay(100);
}
```

### 24.4.2 键盘控制程序设计

如图 24-8 所示，通过对 4 个按键的操作进行中断处理，分别在显示电路中显示其重量、单价和总价。

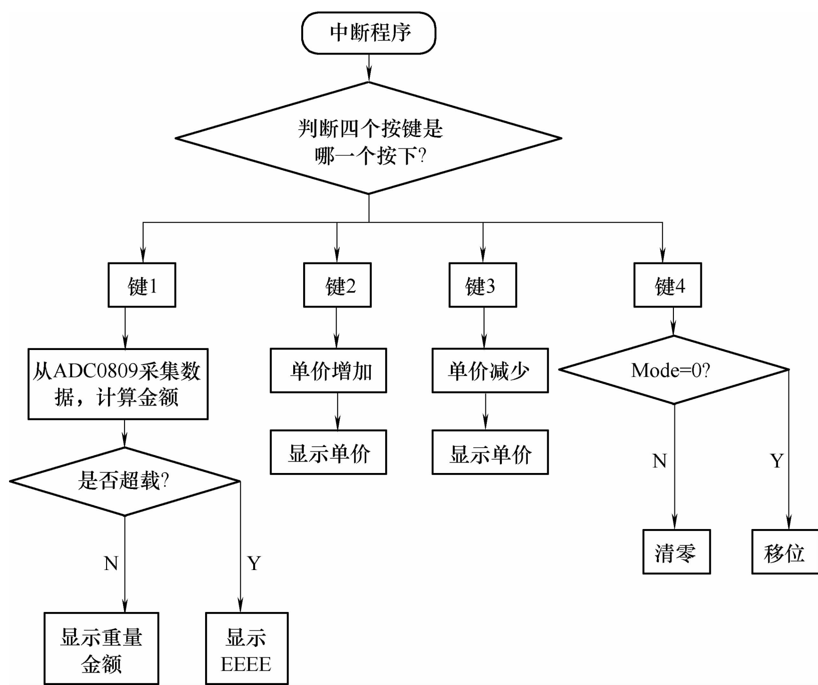



图 24-8 按键程序流程图

其关键程序如下：

```
if(p10 == 0)//监测到按键 1
  {
```

```
        delay(100);
        if(p10 ==0)
      {
          if(mode ==0)
          {   mode =1; transfer(0,10);
          //add code here,start a/d,read in data//
            AD =0; delay(5);
            temp = AD; P2 =0XFF;P0 =0XFF; heavy = temp;
            LOAD =1; CLK =1;
mult = (unsigned long int)(price[0] * 100 + price[1] * 10 + price[2]) * heavy;
            mult = mult/10;
            if((mult >9999)||(heavy >200))
            {
              total[0] =14; total[1] =14;
              total[2] =14; total[3] =14;
            }
            else
            {
              total[0] = mult/1000; total[1] = (mult%1000)/100;
              total[2] = (mult%100)/10; total[3] = mult%10;
            }
            d[0] = temp/100; d[1] = (temp%100)/10;
d[2] = temp%10;   transfer(d[0],0); transfer(d[1],1);transfer(d[2],2);
            transfer(total[0],4);transfer(total[1],5);
            transfer(total[2],6);
            transfer(total[3],7);transfer(d[0],0);
            }
          }
        }
        if(p11 ==0)//检测到按键 2
          {
          delay(100);
          if(p11 ==0)
          {   if(price[price_pos] <9)price[price_pos] ++;
          else price[price_pos] =0; transfer(price[price_pos],price_pos);
          }
        }
        if(p12 ==0)//检测到按键 3
        {
```

```
        delay(100);
        if(p12 ==0)
        {   if(price[price_pos] >0)price[price_pos] --;
          else price[price_pos] =9; transfer(price[price_pos],price_pos);
        }
      }
      if(p13 ==0)//检测到按键4
      {
        delay(100);
        if(p13 ==0)
        {
          if(mode ==1)
          {   mode =0; heavy =0; price_pos =0;
              total[0] =0;total[1] =0;total[2] =0;total[3] =0;
              price[0] =0; price[1] =0; price[2] =0;
              transfer(price[0],0);transfer(price[1],1);transfer(price[2],2);
transfer(total[0],4);transfer(total[1],5);transfer(total[2],6);
transfer(total[3],7);
          }
          else
          {   if(price_pos <2)price_pos ++;
            else price_pos =0;
          }
        }
      }
```

### 24.4.3 显示程序设计

系统调试时，为了更为直观查看系统的数据流向，选择两块单片机对电子秤进行数据控制，其中，主单片机对采集重量信号和按键信号进行处理，从单片机即显示电路实现重量、价格信号的数据显示。显示模块程序流程如图24-9所示。

其显示电路关键代码如下：

```
void int0() interrupt 0 //  对接收到主模块的数据进行处理,在数码管上显示
    EX0 =0;
    prm_1 =0; prm_2 =0;
    for(j =0;j <8;j ++)
      if(shift[j] ==1)prm_1 += (0x01 <<j);
    for(j =8;j <16;j ++)
      if(shift[j] ==1)prm_2 += (0x01 <<(j -8));
    if(prm_2 ==10)
```

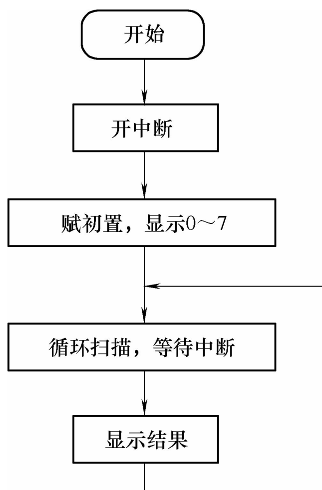


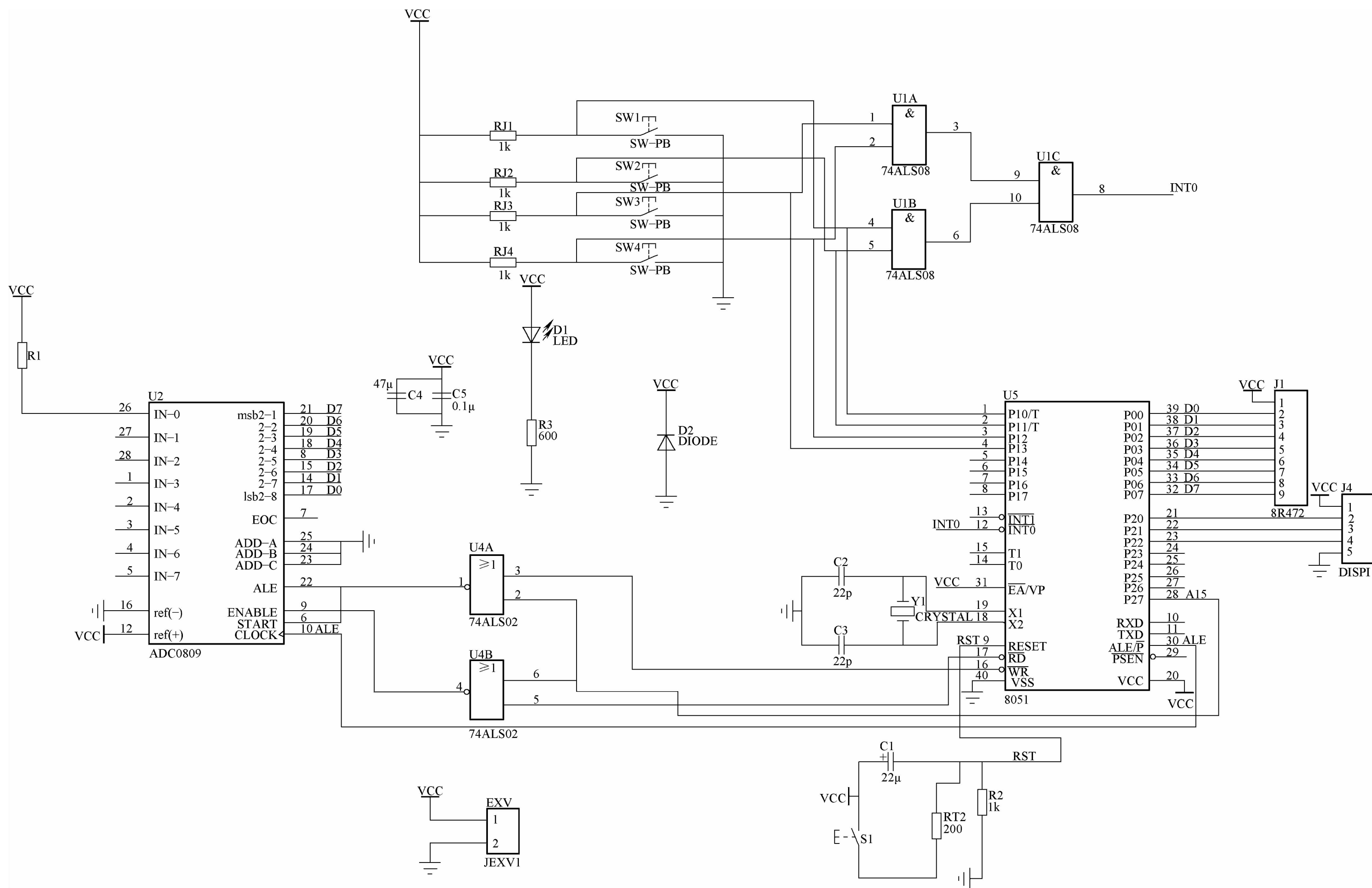

图 24-9　显示模块程序流程图

```
        for(j=0;j<8;j++) pos[j]=0;
    else { pos[prm_2]=1; dat[prm_2]=prm_1; }
    i=0;
    EX0=1;
}
void int1() interrupt 2   //接收主模块传送的数据
{
    EX1=0;
    if(i<16)
    {
        if(DIN==1)shift[i++]=1;
        else shift[i++]=0;
    }
    EX1=1;
}
```

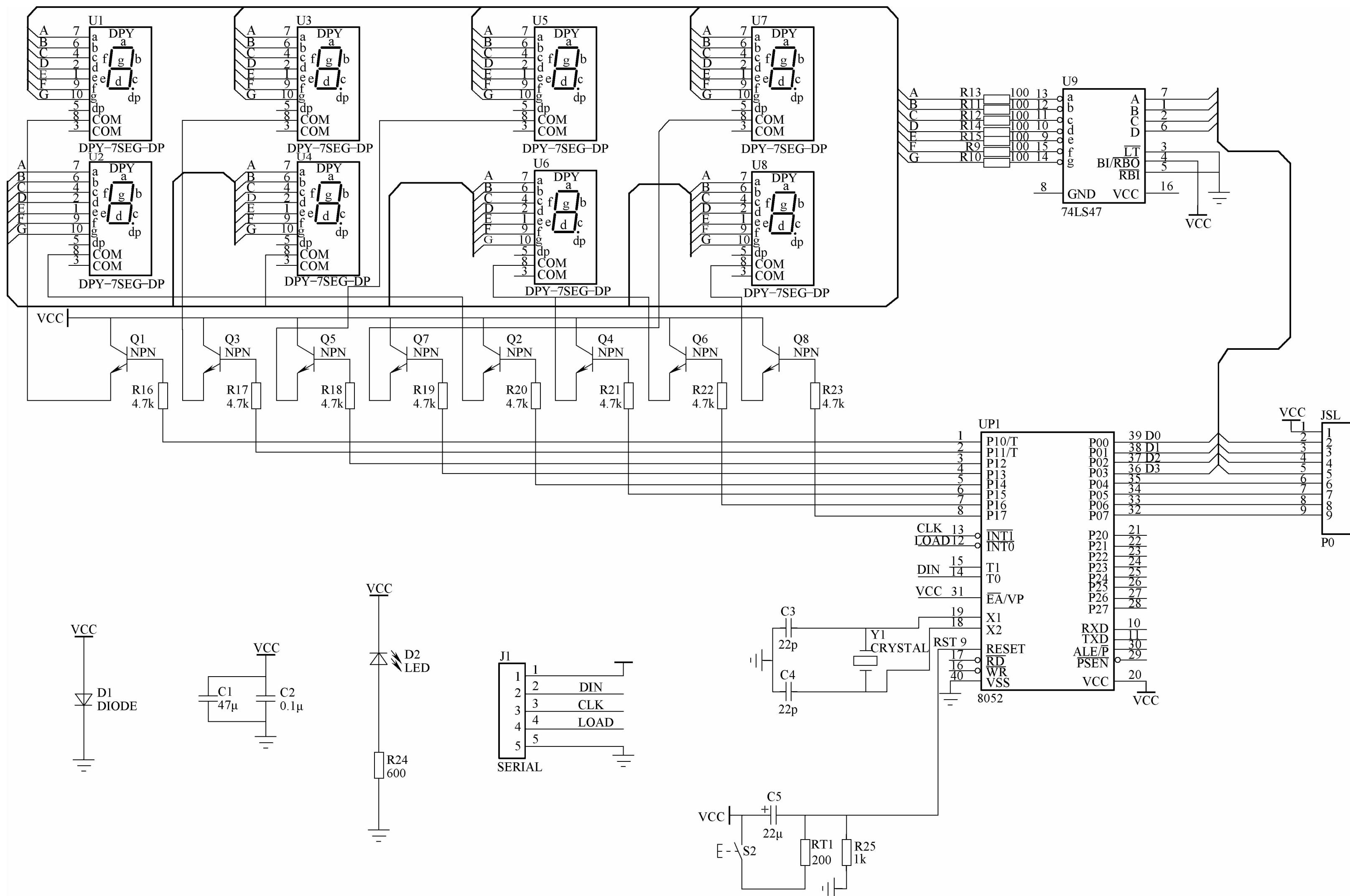
## 24.5　系统调试总结

将信号采集模块和显示模块分开设计电路，然后由数据线进行联通传递采集到的信号，从信号采集模块中将数据经过 A-D 转换后，再由单片机对采集的数据进行处理，传送给显示模块的单片机进一步处理后由数码管显示出重量、单价以及金额，可以让用户比较清楚直观地了解到所需要的信息。

## 附件：系统设计的电路原理图


VCC
RJ1 1k
RJ2 1k
RJ3 1k
RJ4 1k
SW1
SW2
SW3
SW4
SW-PB
U1A
U1B
U1C
&
74ALS08
INT0
D1
LED
R3
600
VCC
47μ
C4
C5
0.1μ
D2
DIODE
R1
U2
IN-0
IN-1
IN-2
IN-3
IN-4
IN-5
IN-6
IN-7
ref(-)
ref(+)
msb2-1
2-2
2-3
2-4
2-5
2-6
2-7
lsb2-8
EOC
ADD-A
ADD-B
ADD-C
ALE
ENABLE
START
CLOCK
ADC0809
D7
D6
D5
D4
D3
D2
D1
D0
10 ALE
U4A
U4B
≥1
74ALS02
U5
P10/T
P11/T
P12
P13
P14
P15
P16
P17
INT1
INT0
T1
T0
EA/VP
X1
X2
RESET
RD
WR
VSS
8051
P00
P01
P02
P03
P04
P05
P06
P07
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
VCC
28 A15
30 ALE
J1
8R472
J4
DISPI
C2
22p
C3
22p
Y1
CRYSTAL
RST
C1
22μ
RT2
200
R2
1k
S1
EXV
JEXV1

附件1　电子秤主控电路

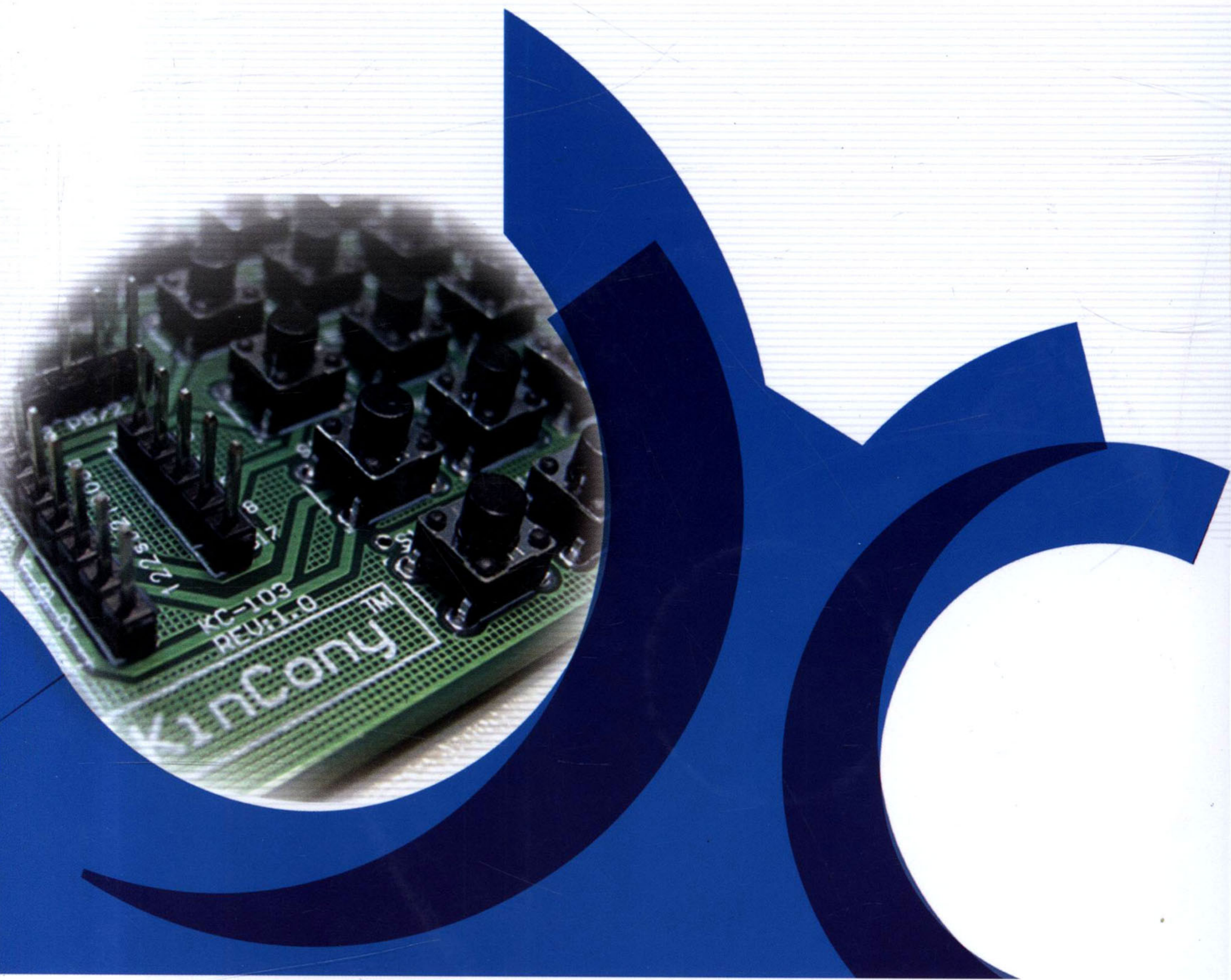

U1
U2
U3
U4
U5
U6
U7
U8
DPY
COM
DPY-7SEG-DP
VCC
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
NPN
R16
R17
R18
R19
R20
R21
R22
R23
4.7k
U9
74LS47
R13
R11
R12
R14
R15
R9
R10
100
BI/RBO
RBI
LT
GND
UP1
8052
P10/T
P11/T
P12
P13
P14
P15
P16
P17
P00
P01
P02
P03
P04
P05
P06
P07
D0
D1
D2
D3
JSL
P0
CLK
LOAD
INT1
INT0
DIN
T1
T0
EA/VP
X1
X2
RESET
RD
WR
VSS
RXD
TXD
ALE/P
PSEN
P20
P21
P22
P23
P24
P25
P26
P27
C3
C4
22p
Y1
CRYSTAL
RST
C5
22μ
RT1
200
R25
1k
S2
D1
DIODE
C1
47μ
C2
0.1μ
D2
LED
R24
600
J1
SERIAL

附件2　电子秤显示电路

# 参 考 文 献

［1］ 徐玮，沈建良．单片机快速入门［M］．北京：北京航空航天大学出版社，2008．
［2］ 徐玮，徐富军，沈建良．C51 单片机高效入门［M］．北京：机械工业出版社，2007．
［3］ 戴佳，戴卫恒，刘博文．51 单片机 C 语言应用程序设计实例精讲［M］．北京：电子工业出版社，2008．
［4］ 谭浩强．C 程序设计［M］．北京：清华大学出版社，1991．
［5］ 王津．微机原理及应用［M］．北京：机械工业出版社，2002．
［6］ 兰吉昌．51 单片机应用设计百例［M］．北京：化学工业出版社，2009．
［7］ 徐新民，肖译，李林功，施竞文．单片机原理与应用［M］．杭州：浙江大学出版社，2007．
［8］ 周兴华．手把手教你学单片机 C 程序设计［M］．北京：北京航空航天大学出版社，2007．
［9］ 马忠梅，籍顺心，张凯，马岩．单片机的 C 语言应用程序设计［M］．北京：北京航空航天大学出版社，2007．
［10］ I Scott MacKenzie．THE 8051MICROCONTROLLER［M］．New Jersey：Prentice-Hall．Inc，2009．
［11］ 赵建领，崔昭霞．精通 51 单片机开发技术与应用实例［M］．北京：电子工业出版社，2012．
［12］ 赵建领，薛园园．零基础学单片机 C 语言程序设计［M］．北京：机械工业出版社，2009．
［13］ 宋戈，黄鹤松，员玉良，蒋海峰．51 单片机应用开发范例大全［M］．北京：人民邮电出版社，2012．
［14］ 贺亮．从零开始学 51 单片机［M］．北京：电子工业出版社，2012．
［15］ 张革新，朴健浩．具有语音播报功能的温湿度测量仪设计［J］．大连民族学院学报：2007．
［16］ 沙占友．集成化智能传感器原理与应用［M］．北京：电子工业出版社，2004．
［17］ 陈钊，郭永彩，高潮，张天华．微环境温湿度智能化测量仪研究［J］．仪器仪表学报：2004，（25）．
［18］ 孙环，滕召胜．基于 SHT10 单片集成传感器温湿度检测模块设计［J］．国外电子测量技术，2006，（25）．
［19］ 范寒柏，陈旭升，李雪梅．基于 ISO4000 系列芯片智能录放系统设计［J］．电子技术应用，2007，（33）．
［20］ sensirion the sensor company．SHT1x/SHT7x Humidity & Temperature Sensor［Z］．Zurich Switzerland，2004（7）．
［21］ 崔晓燕．AT89 系列高性能单片机及其应用［J］．微型机与应用，1996．
［22］ 蔡美琴．MCS-51 单片机原理与应用［M］．北京：高等教育出版社，1992．
［23］ 王建校，杨建国．51 系列单片机及 C51 程序设计［M］．北京：科学出版社，2002．
［24］ 孙涵芳，徐爱卿．MCS-51、96 系列单片机原理及应用［M］．北京：航空航天大学出版社，1996．
［25］ 吴成明．可充电电池及其充电器的发展动向［J］．浙江农村技术师范专科学校学报，1988．
［26］ 周玲．基于单片机控制的智能充电器设计［D］．中国优秀硕士学位论文全文数据库，2007．
［27］ 卢治功，贾治国．基于 MAX1898 的智能充电器设计［J］．电子与电脑，2008，（11）．
［28］ 李国厚，冯启高．基于单片机的智能充电器的设计［J］．仪表技术，2001，（01）．
［29］ 温铁钝，孙键国，张天宏．无线遥控智能插座的设计［J］．测控技术，2003，22（10）．
［30］ Koenig，Volker and You，Eugene．Wireless Control Systems for Industrial Applications［J］．Heating Piping Air Conditioning HPAC Engineering，2009．
［31］ Wang Ning，Zhang Naiqian，Wang Maohua．Wireless sensors in agriculture and food industry recent devel-

opment and future perspective [J]. Computers and Electronics in Agriculture, 2006, 50 (1): 1-14.
[32] Lin Jzau-Sheng, Liu Chun-Zu. A monitoring system based on wireless sensor network and an SoC platform in precision agriculture [J]. Communication Technology International Conference on Communication Technology, 2008, 29 (1): 101-104.
[33] 郭志伟，张云伟，李霜. 基于 GSM 的农田气象信息远程监控系统设计 [J]. 农业机械学报，2009，40 (3): 161-166.
[34] Sahota Herman, Kumar Ratnesh, Kamal Ahmed, Jing Huang. An energy-efficient wireless sensor network for precision agriculture [J]. IEEE Symposium on Computers and Communications, 2010, 22 (7): 347-350.
[35] Xiao Lei, Guo Lejiang. The realization of precision agriculture monitoring system based on wireless sensor network [J]. Computer and Communication Technologies in Agriculture Engineering, 2010, 11 (9): 89-92.
[36] 夏华. 无线通信模块设计与物联网应用开发 [M]. 北京：电子工业出版社，2011.
[37] 周立功. 增强型 80C51 单片机速成与实战 [M]. 北京：航空航天大学出版社，2003.
[38] 李忠国，陈刚. 单片机应用技能实训 [M]. 北京：人民邮电出版社，2010.
[39] 徐科军，马修水，李晓林. 传感器与检测技术 [M]. 北京：电子工业出版社，2008.
[40] 周越. 单片机应用技术 [M]. 北京：中国水利水电出版社，2009.
[41] 罗平. 格力牌 FSA—35B 遥控台地扇原理与维修 [J]. 家电检修技术，2000，(06).
[42] 刘进山. 基于 MCS-51 电风扇智能调速器的设计 [J]. 电子质量，2004，(10).
[43] 谢志平. 基于单片机控制的智能温控风扇 [J]. 中国新技术新产品，2011，(02).




地址：北京市百万庄大街22号
邮政编码：100037

电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

ISBN 978-7-111-47690-0
ISBN 978-7-89405-661-0
策划编辑◎林春泉

ISBN 978-7-111-47690-0



定价：59.00元(含1CD)